debug ipv6 inspect through debug local-ack state

debug ipv6 inspect

To display messages about Cisco IOS firewall events, use the **debug ipv6 inspect** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipv6 inspect {**function-trace**| **object-creation**| **object-deletion**| **events**| **timers**| **protocol**| **detailed**}

no debug ipv6 inspect detailed

Syntax Description

function-trace	Displays messages about software functions called by the Cisco IOS firewall.
object-creation	Displays messages about software objects being created by the Cisco IOS firewall. Object creation corresponds to the beginning of Cisco IOS firewall-inspected sessions.
object-deletion	Displays messages about software objects being deleted by the Cisco IOS firewall. Object deletion corresponds to the closing of Cisco IOS firewall-inspected sessions.
events	Displays messages about Cisco IOS firewall software events, including information about Cisco IOS firewall packet processing.
timers	Displays messages about Cisco IOS firewall timer events such as when a Cisco IOS firewall idle timeout is reached.
protocol	Displays messages about Cisco IOS firewall-inspected protocol events, including details about the protocol's packets.
detailed	Use this form of the command in conjunction with other Cisco IOS firewall debugging commands. This causes detailed information to be displayed for all the other enabled Cisco IOS firewall debugging.

Command Default

None

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(2)T	This command was introduced
12.0(22)S	This command was integrated into Cisco IOS Release 12.0(22)S.
12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Command History

12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.

Examples

The following example enables the display of messages about Cisco IOS firewall events:

```
debug ipv6 inspect
```

Related Commands

Command	Description
ipv6 inspect audit-trail	Turns on CBAC audit trail messages, which are displayed on the console after each Cisco IOS firewall session closes.
ipv6 inspect name	Defines a set of ipv6 inspection rules.
show ipv6 inspect	Displays CBAC configuration and session information.

debug ipv6 mfib

To enable debugging output on the IPv6 Multicast Forwarding Information Base (MFIB), use the **debug ipv6 mfib** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipv6 mfib [**vrf** *vrf-name*] [*group-name*| *group-address*] [**adjacency**| **db**| **fs**| **init**| **interface**| **mrib** **[detail]**| **nat**| **pak**| **platform**| **ppr**| **ps**| **signal**| **table**]

no debug ipv6 mfib

Syntax Description

vrf *vrf-name*	(Optional) Specifies a virtual routing and forwarding (VRF) configuration.
group-name | *group-address*	(Optional) IPv6 address, name, or interface of the multicast group as defined in the Domain Name System (DNS) hosts table.
adjacency	(Optional) Enables debugging output for adjacency management activity.
db	(Optional) Enables debugging output for route database management activity.
fs	(Optional) Enables debugging output for fast switching activity.
init	(Optional) Enables debugging output for initialization or deinitialization activity.
interface	(Optional) Enables debugging output for IPv6 MFIB interfaces.
mrib	(Optional) Enables debugging output for communication with the MRIB.
detail	(Optional) Enables detailed debugging output regarding the MRIB.
nat	(Optional) Enables debugging output for Network Address Translation (NAT) events associated with all tables.
pak	(Optional) Enables debugging output for packet forwarding activity.
platform	(Optional) Enables debugging output related to the hardware platform use of application program interfaces (APIs).

ppr	(Optional) Enables debugging output for packet preservation events.
ps	(Optional) Enables debugging output for process-level-only packet forwarding activity.
signal	(Optional) Enables debugging output for activity regarding MFIB data-driven signaling to routing protocols.
table	(Optional) Enables debugging output for IPv6 MFIB table activity.

Command Modes

Privileged EXEC

Release	Modification
12.3(2)T	This command was introduced.
12.2(18)S	This command was integrated into Cisco IOS Release 12.2(18)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
Cisco IOS XE Release 2.1	This command was introduced on Cisco ASR 1000 series routers.
12.2(33)SRE	The **detail** keyword was added.
15.1(1)T	The **detail** keyword was added.
15.1(4)M	The **vrf** *vrf-name* keyword and argument were added.

Usage Guidelines

If no keywords are used, all IPbv6 MFIB activity debugging output is displayed.

Examples

The following example enables debugging output for adjacency management activity on the IPv6 MFIB:

```
Router# debug ipv6 mfib adjacency
```

debug ipv6 mld

To enable debugging on Multicast Listener Discovery (MLD) protocol activity, use the **debug ipv6 mld**command in privileged EXEC mode. To restore the default value, use the **no** form of this command.

debug ipv6 mld [*group-name*| *group-address*| *interface-type*]

no debug ipv6 mld [*group-name*| *group-address*| *interface-type*]

Cisco IOS Release 12.0(26)S

debug ipv6 mld [**group** *group-name*| *group-address*| **interface** *interface-type*]

no debug ipv6 mld [**group** *group-name*| *group-address*| **interface** *interface-type*]

Syntax Description

group-name | *group-address*or **group** *group-name* | *group-address*	(Optional) IPv6 address or name of the multicast group.
interface-type or **interface** *interface-type*	(Optional) Interface type. For more information, use the question mark (**?**) online help function.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(2)T	This command was introduced.
12.2(18)S	This command was integrated into Cisco IOS Release 12.2(18)S.
12.0(26)S	This command was integrated into Cisco IOS Release 12.0(26)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(25)SG	This command was integrated into Cisco IOS Release 12.2(25)SG.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
Cisco IOS XE Release 2.1	This command was introduced on Cisco ASR 1000 Series Routers.

Usage Guidelines

This command helps discover whether the MLD protocol activities are working correctly. In general, if MLD is not working, the router process never discovers that there is a host on the network that is configured to receive multicast packets.

The messages displayed by the **debug ipv6 mld**command show query and report activity received from other routers and hosts. Use this command in conjunction with **debug ipv6 pim** to display additional multicast activity, to learn more information about the multicast routing process, or to learn why packets are forwarded out of particular interfaces.

Examples

The following example enables debugging on MLD protocol activity:

```
Router# debug ipv6 mld
```

Related Commands

Command	Description
debug ipv6 pim	Enables debugging on PIM protocol activity.

debug ipv6 mld explicit

To display information related to the explicit tracking of hosts, use the **debug ipv6 mld explicit**command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug ipv6 mld explicit [*group-name*| *group-address*]

no debug ipv6 mld explicit [*group-name*| *group-address*]

Syntax Description

group-name | *group-address*	(Optional) IPv6 address or name of the multicast group.

Command Default

Debugging for the explicit tracking of hosts is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(7)T	This command was introduced.
12.2(25)S	This command was integrated into Cisco IOS Release 12.2(25)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(25)SG	This command was integrated into Cisco IOS Release 12.2(25)SG.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
Cisco IOS XE Release 2.1	This command was introduced on Cisco ASR 1000 Series Routers.

Usage Guidelines

When the optional *group-name* or *group-address* argument is not used, all debugging information is displayed.

Examples

The following example shows how to enable information to be displayed about the explicit tracking of hosts. The command output is self-explanatory:

```
Router# debug ipv6 mld explicit
00:00:56:MLD:ET host FE80::A8BB:CCFF:FE00:800 report for FF05::6 (0 srcs) on Ethernet1/0
00:00:56:MLD:ET host FE80::A8BB:CCFF:FE00:800 switch to exclude for FF05::6 on Ethernet1/0
00:00:56:MLD:ET MRIB modify for (*,FF05::6) on Ethernet1/0 new 100, mdf 100
```

debug ipv6 mld ssm-map

To display debug messages for Source Specific Multicast (SSM) mapping related to Multicast Listener Discovery (MLD), use the **debug ipv6 mld ssm-map**command in privileged EXEC mode. To disable debug messages for SSM mapping, use the **no** form of this command.

debug ipv6 mld ssm-map [*source-address*]

no debug ipv6 mld ssm-map [*source-address*]

Syntax Description

source-address	(Optional) Source address associated with an MLD membership for a group identified by the access list.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(18)SXE	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

Consult Cisco technical support before using this command.

Examples

The following example allows debugging information for SSM mapping to be displayed:

```
Router# debug ipv6 mld ssm-map
```

Related Commands

Command	Description
ipv6 mld ssm-map enable	Enables the SSM mapping feature for groups in the configured SSM range
ipv6 mld ssm-map query dns	Enables DNS-based SSM mapping.
ipv6 mld ssm-map static	Configures static SSM mappings.
show ipv6 mld ssm-map	Displays SSM mapping information.

debug ipv6 mobile

To enable the display of debugging information for Mobile IPv6, use the **debug ipv6 mobile**command in privileged EXEC mode.

debug ipv6 mobile {**binding-cache**| **forwarding**| **home-agent**| **registration**}

Syntax Description

binding-cache	Events associated with the binding cache.
forwarding	Events associated with forwarding (tunneling) packets for which the router is acting as home agent.
home-agent	Events associated with the home agent, Dynamic Home Address Agent Discovery (DHAAD), Mobile prefix discovery (MPD), and generic home agent (HA) debugging and binding acknowledgments.
registration	Events associated with binding updates that are registrations.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(14)T	This command was introduced.

Usage Guidelines

The **debug ipv6 mobile** command enables the display of selected debugging information. You may use multiple command lines to enable concurrent debugging of multiple classes of information.

Examples

In the following example, debugging information is displayed for binding updates processing:

```
Router# debug ipv6 mobile registration
```

Related Commands

Command	Description
binding	Configures binding options for the Mobile IPv6 home agent feature in home-agent configuration mode.
ipv6 mobile home-agent (global configuration)	Enters home agent configuration mode.

Command	Description
ipv6 mobile home-agent (interface configuration)	Initializes and start the IPv6 Mobile home agent on a specific interface.
ipv6 mobile home-agent preference	Configures the home agent preference value on the interface.

debug ipv6 mobile mag

To debug the Mobile Access Gateway (MAG) application programming interface (API), information, or events, use the **debug ipv6 mobile mag** command in privileged EXEC mode. To disable display of the debugging output, use the **no** form of this command.

debug ipv6 mobile mag {**api**| **events**| **info**}

no debug ipv6 mobile mag {**api**| **events**| **info**}

Syntax Description

api	Enables API-specific debug events.
events	Enables all events occurring within the Local Mobility Anchor (LMA) and the MAG.
info	Provides debug information within the Proxy Mobile IPv6 (PMIPV6) module.

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.4S	This command was introduced.
15.2(4)M	This command was integrated into Cisco IOS Release 15.2(4)M.

Usage Guidelines

Use the **debug ipv6 mobile mag events** command to enable events occurring within the LMA and MAG. The following table lists the common causes for Proxy Binding Update (PBU) rejections:

PBU Reject Status	Description
PMIPV6_BA_ACCEPTED	The PBU is accepted.
GRE_KEY_OPTION_NOT_REQUIRED	The PBU is processed successfully but the GRE encapsulation and GRE keys are not required.
PMIPV6_BA_UNSPEC_FAIL	The PBU is rejected for an unspecified reason.
PMIPV6_BA_ADMIN_FAIL	The PBU is rejected due to administrative reasons.

PBU Reject Status	Description
PMIPV6_BA_RESOURCE_FAIL	The PBU is rejected due to insufficient resources.
PMIPV6_BA_HM_REG_FAIL	The PBU is rejected because it has an unsupported home registration.
PMIPV6_BA_HM_SUBNET_FAIL	The PBU is rejected because the current subnet is not the home subnet.
PMIPV6_BA_BAD_SEQ_FAIL	The PBU is rejected because the sequence number is out of the specified range.
PMIPV6_BA_CHANGE_FAIL	The PBU is rejected because the registration type has changed.
PMIPV6_BA_AUTH_FAIL	The PBU is rejected because the authorization has failed.
PROXY_REG_NOT_ENABLED	The PBU is rejected because the registration of the proxy is not enabled for the mobile node.
NOT_LMA_FOR_THIS_MOBILE_NODE	The PBU is rejected because the current Local Mobility Anchor (LMA) is not the appropriate LMA for the mobile node.
MAG_NOT_AUTHORIZED_FOR_ PROXY_REG	The PBU is rejected because the Mobile Access Gateway (MAG) is not authorized to send PBUs.
NOT_AUTHORIZED_FOR_HNP	The PBU is rejected because it is not authorized for the Home Network Prefix (HNP).
TIMESTAMP_MISMATCH	The PBU is rejected because it has an invalid timestamp value.
TIMESTAMP_LOWER_THAN_ PREV_ACCEPTED	This PBU is rejected because the timestamp value is lower than the previously accepted value.
MISSING_HNP_OPTION	The PBU is rejected because it is the Home Network Prefix (HNP) option.
BCE_PBU_PREFIX_SET_DO _NOT_MATCH	The PBU is rejected because the Home Network Prefixes (HNPs) that are received in the PBU do not match with the Binding Cache Entry (BCE).
MISSING_MN_IDENTIFIER_OPTION	The PBU is rejected because the mobile node identifier option is missing.
MISSING_HANDOFF_INDICATOR_ OPTION	The PBU is rejected because the Handoff Indicator is missing.

Examples

The following is sample output from the **debug ipv6 mobile mag api** command displays the APIs that are called during the call setup flow:

```
Device# debug ipv6 mobile mag api

07:52:08.051: MIP_PDL_API: pmipv6_pdl_get_att API Called
07:52:08.051: [PMIPV6_BINDING_API]: pmipv6_get_binding API called
07:52:08.051: [PMIPV6_BINDING_API]: pmipv6_get_binding API called
07:52:08.051: [PMIPV6_MAG_API]: mag_bul_do_state_transition API called
07:52:08.051: [PMIPV6_MAG_API]: pmipv6_mag_bul_null_state_hndlr API called
07:52:08.051: [PMIPV6_MAG_API]: pmipv6_mag_bul_null_state_exit API called
07:52:08.051: [PMIPV6_MAG_API]: pmipv6_mag_bul_init_state_entry API called
07:52:08.051: [PMIPV6_BINDING_API]: pmipv6_add_binding_entry API called
07:52:08.051: MIP_PDL_API: pmipv6_pdl_get_timestamp API Called
07:52:08.053: [PMIPV6_MAG_API]: pmipv6_mag_should_handle_pkt called
07:52:08.053: [PMIPV6_MAG_API]: pmipv6_mag_message_handler called
07:52:08.053: [PMIPV6_BINDING_API]: pmipv6_get_binding API called
07:52:08.053: [PMIPV6_BINDING_API]: pmipv6_get_binding API called
07:52:08.053: [PMIPV6_MAG_API]: mag_bul_do_state_transition API called
07:52:08.053: [PMIPV6_MAG_API]: pmipv6_mag_bul_init_state_hndlr API called
07:52:08.053: [PMIPV6_MAG_API]: pmipv6_mag_bul_init_state_exit API called
07:52:08.053: MIP_PDL_API: pmipv6_pdl_create_vintf API Called
16 07:52:08.054: MIP_PDL_API: pmipv6_pdl_set_ip4address API Called
16 07:52:08.054: MIP_PDL_API: pmipv6_pdl_set_macaddr API Called
16 07:52:08.054: MIP_PDL_API: mip_pdl_setupv4_route API Called
07:52:08.054: MIP_PDL_API: mip_pdl_setupv6_tunnel API Called
07:52:08.054: %LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel0, changed state to
down
07:52:08.054: MIP_PDL_API: mip_pdl_get_handle_for_tunnel API Called
07:52:08.054: MIP_PDL_API: mip_pdl_populate_rtunnel API Called
07:52:08.054: MIP_PDL_API: mip_pdl_get_handle_for_tunnel API Called
07:52:08.055: [PMIPV6_BINDING_API]: pmipv6_update_binding_key API called
07:52:08.055: [PMIPV6_MAG_API]:  pmipv6_mag_bul_active_state_entry API called
```

The following is sample output from the **debug ipv6 mobile mag events** command:

```
Device# debug ipv6 mobile mag events

PMIPv6 MAG Event debug is turned on
```

The following line shows that the DHCP Discover trigger is received from the mobile node (MN):

```
07:48:31.638: [PMIPV6_MAG_EVENT]: Trigger request received (DHCP Discover trigger) from
(example3@example.com)
```

The following line shows the MAG machine state change. A new MN attaches to the MAG and the state changes from NULL to INIT:

```
07:48:31.638: [PMIPV6_MAG_EVENT]: Event received New MN intf attached in state: NULL, new
state: INIT
```

The following line shows that the Proxy Binding Update (PBU) message is sent from a MAG to an MN:

```
07:48:31.638: [PMIPV6_MAG_EVENT]: PBU message sent
```

The following lines show that the Proxy Binding Acknowledgment (PBA) is received from the LMA for the MN. The incoming parameters are link layer identifier (lli) length, value, and access technology type (att). The status 0 indicates success.

```
07:48:31.639: [PMIPV6_MAG_EVENT]: message received: PBA
07:48:31.639: [PMIPV6_MAG_EVENT]: PBA: nai(example3@example.com),nai len: 14, lli
(aabb.cc00.ce00), ll len: 16, att:3, status:0
```

The following line shows that the refresh timer has started:

```
07:48:31.639: [PMIPV6_MAG_EVENT]: Starting Refresh timer, period (300000)
```

The following lines show that a v4 route is added to the MN, which has a new address assigned. A new v6 tunnel is created and a reverse tunnel entry is added for the MN.

```
07:48:31.640: [PMIPV6_MAG_EVENT]: Adding V4 route, address (0x11110103), Prefix len (24),
handle: (GigabitEthernet0/0/0)
!
07:48:31.640: [PMIPV6_MAG_EVENT]: Adding V6 Tunnel, Handle (Tunnel1), mode: (IPV6_IN_IPV6)
07:48:31.641: [PMIPV6_MAG_EVENT]: Populating  Reverse V4 Tunnel entry, l2 address
(0xaabb.cc00.ce00), ipv4 add: 0x11110103 phy handle: (GigabitEthernet0/0/0)
```

The following is sample out from **debug ipv6 mobile mag info** command:

```
Device# debug ipv6 mobile mag info

PMIPv6 MAG INFO debug is turned on
```

The following lines show that the new binding is created and added to the AV tree:

```
07:50:31.714: [PMIPV6_PDB_INFO]: MN entry example3@example.com found in hashset
07:50:31.714: [PMIPV6_BINDING_INFO]: binding added New NAI AVL node created
```

The following line provides more information about the PBUs that are sent:

```
07:50:31.714: [PMIPV6_MAG_INFO]: PBU message nai(example3@example.com), nai len: 14, hoa(0),
 att(3) llid(aabb.cc00.ce00) , ll len: 16
```

The following line shows that a binding for the MN using the Network Access Identifier (NAI) example3@example.com is found:

```
07:50:31.717: [PMIPV6_BINDING_INFO_KEY]: Keytype as NAI. NAI: example3@example.com
07:50:31.717: [PMIPV6_BINDING_INFO]: binding found on NAI tree
```

The following line shows that a virtual interface is created in the MAG and assigned the MAC address aaaa.aaaa.aaaa:

```
07:50:31.717: [PMIPV6_MAG_EVENT]: Creating virtual interface handle (IFNAME_PMIP_VIF4)
07:50:31.717: [PMIPV6_MAG_INFO]: Setting Mac Address (aaaa.aaaa.aaaa) on (IFNAME_PMIP_VIF4)
The following line shows that a route for the MN is added in the MAG:
07:50:31.717: MIP_PDL_INFO: Successfully added route 10.10.1.4/24 to GigabitEthernet0/0/0
07:50:31.717: MIP_PDL_INFO: Route via: GigabitEthernet0/1/0 (IPv6)
The following line shows that a tunnel is created with a source address and a destination
address:
07:50:31.718: MIP_PDL_INFO: Tunnel0 (IPv6) created with src 2000::4 dst 2001::2
07:50:31.718: MIP_PDL_INFO: Rev. Tunnel acl entry added for subnet (10.10.0.0)
```

Related Commands

Command	Description
ipv6 mobile pmipv6-mag	Configures the MAG for the PMIPV6 domain.

debug ipv6 mobile networks

To display debugging messages for IPv6 mobile networks, use the **debug ipv6 mobile networks**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipv6 mobile networks

no debug ipv6 mobile networks

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(20)T	This command was introduced.

Usage Guidelines

The **debug ipv6 mobile networks**command enables the display of selected debugging information.

Examples

The following example shows how to enable the display of debugging messages for IPv6 mobile networks:

```
Router# debug ipv6 mobile networks
```

Related Commands

Command	Description
ipv6 mobile router	Enables IPv6 NEMO functionality on a router and places the router in IPv6 mobile router configuration mode.

debug ipv6 mobile packets

To debug the proxy mobile IPv4 or IPv6 packets, use the **debug ipv6 mobile packets** command in privileged EXEC mode. To disable the debugging output, use the **no** form of this command.

debug ipv6 mobile packets

no debug ipv6 mobile packets

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.4S	This command was introduced.
15.2(4)M	This command was integrated into Cisco IOS Releases 15.2(4)M.

Examples

The following is sample output from the **debug ipv6 mobile packets** command:

```
Device# debug ipv6 mobile packets

PMIPv6 PKT debug is turned on
```

The following lines show the newly allocated packet size and the inner packet details:

```
07:51:17.693: [PMIPv6-MM]:Allocated packet of size 164 with tlv length 84
07:51:17.693: [PMIPV6_MM] Sending UDP Packet, src: 0x2020202, dst: 0x6060602, sport: 5436,
 dport:5436
```

The following lines shows the mobility options, the value, and the length:

```
07:51:17.693: [PMIPV6_MM] NAI option included len 14
!
2A986107E0:                4D 4E334063 6973636F         example3@example
2A986107F0: 2E636F6D 1702                          .com..
07:51:17.693:
07:51:17.693: [PMIPV6_MM] HI option included len 2 val 4
07:51:17.694: [PMIPV6_MM] ATT option included len 2 val 3
07:51:17.694: [PMIPV6_MM] TIMESTAMP option included len 8 value 3517199477
07:51:17.694: [PMIPV6_MM] LLI option included len 16
!
2A98610810: 61616262 2E636330 302E6365 30300100  aabb.cc00.ce00..
2A98610820: 24                                    $
07:51:17.694:
07:51:17.694: [PMIPV6_MM] V4HOAREQ option included len 6 val 0.0.0.0
07:51:17.694: [PMIPV6_MM] V4DFT_RTR option included len 6 val 0.0.0.0
07:51:17.694: **** Dumping the TLVs ****
```

```
!
2A986107E0: 01020000 080E014D 4E334063 6973636F  .......example3@example
2A986107F0: 2E636F6D 17020004 18020003 01001B08  .com............
2A98610800: 00000000 D1A43475 01020000 19100000  ....Q$4u........
2A98610810: 61616262 2E636330 302E6365 30300100  aabb.cc00.ce00..
2A98610820: 24060000 00000000 26060000 00000000  $.......&.......
2A98610830: 01020000                             ....
07:51:17.694:
07:51:17.695: [PMIPV6_MM] NAI option received len 14
!
2A97DBE560:       4D 4E334063 6973636F 2E636F6D     example3@example.com
2A97DBE570: 0017                                 ..
07:51:17.696:
07:51:17.696: [PMIPV6_MM] HI option received len 2 val 4
07:51:17.696: [PMIPV6_MM] ATT option received len 2 val 3
07:51:17.696: [PMIPV6_MM] TIMESTAMP option received len 8 value 3517199477
07:51:17.696: [PMIPV6_MM] LLI option received len 16
!
2A97DBE580:                              61616262             aabb
2A97DBE590: 2E636330 302E6365 30300100 00        .cc00.ce00...
07:51:17.696:
07:51:17.696: [PMIPV6_MM] V4HOAREPLY option received len 6 val 10.10.1.5
07:51:17.696: [PMIPV6_MM] V4DFT_RTR option received len 6 val 10.10.1.1
```

The following lines show the dump of the packet with all the Type Length Values (TLVs):

```
07:51:17.696: **** Dumping the TLVs ****
!
2A97DBE550:                              01020000             ....
2A97DBE560: 080E014D 4E334063 6973636F 2E636F6D  ...example3@example.com
2A97DBE570: 00170200 04180200 03001B08 00000000  ................
2A97DBE580: D1A43475 01020000 19100000 61616262  Q$4u........aabb
2A97DBE590: 2E636330 302E6365 30300100 00000000  .cc00.ce00......
2A97DBE5A0: 00000000 00000000 00000000 00000000  ................
2A97DBE5B0: 25060060 11110105 26060000 11110101  %..`....&.......
2A97DBE5C0:
07:51:17.696:
```

Related Commands

Command	Description
ipv6 mobile pmipv6-mag	Configures the MAG for the PMIPv6 domain.

debug ipv6 mobile router

To display debugging messages for the IPv6 mobile router, use the **debug ipv6 mobile router** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipv6 mobile router [detail]

no debug ipv6 mobile router

Syntax Description

detail	(Optional) Displays detailed mobile router debug messages.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(20)T	This command was introduced.

Usage Guidelines

The IPv6 mobile router operations can be debugged. The following conditions trigger debugging messages:

- Agent discovery
- Registration
- Mobile router state change
- Routes and tunnels created or deleted
- Roaming information

Debugging messages are prefixed with "MobRtr," and detail messages are prefixed with "MobRtrX."

Examples

The following example shows how to enable the display of debugging messages for the IPv6 mobile router:

```
Router# debug ipv6 mobile router
```

Related Commands

Command	Description
ipv6 mobile router	Enables IPv6 NEMO functionality on a router and places the router in IPv6 mobile router configuration mode.

debug ipv6 mrib client

To enable debugging on Multicast Routing Information Base (MRIB) client management activity, use the **debug ipv6 mrib client**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipv6 mrib [**vrf** *vrf-name*] **client**

no debug ipv6 mrib client

Syntax Description

vrf *vrf-name*	(Optional) Specifies a virtual routing and forwarding (VRF) configuration.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(2)T	This command was introduced.
12.2(18)S	This command was integrated into Cisco IOS Release 12.2(18)S.
12.0(26)S	This command was integrated into Cisco IOS Release 12.0(26)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(25)SG	This command was integrated into Cisco IOS Release 12.2(25)SG.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
Cisco IOS XE Release 2.1	This command was introduced on Cisco ASR 1000 Series Routers.
15.1(4)M	The **vrf** *vrf-name* keyword and argument were added.

Usage Guidelines

The **debug ipv6 mrib client** command is used to display the activity in the MRIB associated with clients such as Protocol Independent Multicast (PIM) and Multicast Listener Discovery (MLD). If you are having difficulty with your client connections, use this command to display new clients being added and deleted.

The **debug ipv6 mrib client** command also displays information on when a new client is added to or deleted from the MRIB, when a client connection is established or torn down, when a client binds to a particular MRIB table, and when a client is informed that there are updates to be read.

Examples

The following example enables debugging on MRIB client management activity:

```
Router# debug ipv6 mrib client
```

Related Commands

Command	Description
debug ipv6 mrib route	Displays MRIB routing entry-related activity.

debug ipv6 mrib io

To enable debugging on Multicast Routing Information Base (MRIB) I/O events, use the **debug ipv6 mrib io**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipv6 mrib [**vrf** *vrf-name*] **io**

no debug ipv6 mrib io

Syntax Description

vrf *vrf-name*	(Optional) Specifies a virtual routing and forwarding (VRF) configuration.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(2)T	This command was introduced.
12.2(18)S	This command was integrated into Cisco IOS Release 12.2(18)S.
12.0(26)S	This command was integrated into Cisco IOS Release 12.0(26)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(25)SG	This command was integrated into Cisco IOS Release 12.2(25)SG.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
Cisco IOS XE Release 2.1	This command was introduced on Cisco ASR 1000 Series Routers.
15.1(4)M	The **vrf** *vrf-name* keyword and argument were added.

Usage Guidelines

Use the **debug ipv6 mrib io** command to display information on when clients open and close MRIB I/O connections, when MRIB entry and interface updates are received and processed from clients, and when MRIB entry and interface updates are sent to clients.

Examples

The following example enables debugging on MRIB I/O events:

```
Router# debug ipv6 mrib io
```

debug ipv6 mrib proxy

To enable debugging on multicast routing information base (MRIB) proxy activity between the route processor and line cards on distributed router platforms, use the **debug ipv6 mrib proxy**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipv6 mrib proxy

no debug ipv6 mrib proxy

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(26)S	This command was introduced.
12.3(4)T	This command was integrated into Cisco IOS Release 12.3(4)T.
12.2(25)S	This command was integrated into Cisco IOS Release 12.2(25)S.
12.2(25)SG	This command was integrated into Cisco IOS Release 12.2(25)SG.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
Cisco IOS XE Release 2.1	This command was introduced on Cisco ASR 1000 Series Routers.

Usage Guidelines

Use the **debug ipv6 mrib proxy** command to display information on connections that are being opened and closed and on MRIB transaction messages that are being passed between the route processor and line cards.

Examples

The following example enables debugging on MRIB proxy events:

```
Router# debug ipv6 mrib proxy
```

debug ipv6 mrib route

To display information about Multicast Routing Information Base (MRIB) routing entry-related activity, use the **debug ipv6 mrib route**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipv6 mrib [**vrf** *vrf-name*] **route** [*group-name*| *group-address*]

no debug ipv6 mrib route

Syntax Description

vrf *vrf-name*	(Optional) Specifies a virtual routing and forwarding (VRF) configuration.
group-name | *group-address*	(Optional) IPv6 address or name of the multicast group.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(2)T	This command was introduced.
12.2(18)S	This command was integrated into Cisco IOS Release 12.2(18)S.
12.0(26)S	This command was integrated into Cisco IOS Release 12.0(26)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(25)SG	This command was integrated into Cisco IOS Release 12.2(25)SG.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
Cisco IOS XE Release 2.1	This command was introduced on Cisco ASR 1000 Series Routers.
15.1(4)M	The **vrf** *vrf-name* keyword and argument were added.

Usage Guidelines

This command displays update information related to the route database made by MRIB clients, which is then redistributed to the clients.

Use this command to monitor MRIB route activity when discontinuity is found between the MRIB and the client database or between the individual client databases.

Examples

The following example enables the display of information about MRIB routing entry-related activity:

```
Router# debug ipv6 mrib route
```

Related Commands

Command	Description
show ipv6 mrib client	Displays information about the MRIB client management activity.

debug ipv6 mrib table

To enable debugging on Multicast Routing Information Base (MRIB) table management activity, use the **debug ipv6 mrib table**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipv6 mrib [**vrf** *vrf-name*] **table**

no debug ipv6 mrib table

Syntax Description

vrf *vrf-name*	(Optional) Specifies a virtual routing and forwarding (VRF) configuration.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(2)T	This command was introduced.
12.2(18)S	This command was integrated into Cisco IOS Release 12.2(18)S.
12.0(26)S	This command was integrated into Cisco IOS Release 12.0(26)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(25)SG	This command was integrated into Cisco IOS Release 12.2(25)SG.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
Cisco IOS XE Release 2.1	This command was introduced on Cisco ASR 1000 Series Routers.
15.1(4)M	The **vrf** *vrf-name* keyword and argument were added.

Usage Guidelines

Use the **debug ipv6 mrib table** command to display information on new MRIB tables being added and deleted.

Examples

The following example enables debugging on MRIB table management activity:

```
Router# debug ipv6 mrib table
```

debug ipv6 multicast aaa

To enable debugging of authentication, authorization, and accounting (AAA) events related to IPv6 multicast routing, use the **debug ipv6 multicast aaa** command in privileged EXEC mode. To disable debugging of events, use the **no** form of this command.

debug ipv6 multicast aaa {**detail | error | verbose**}

no debug ipv6 multicast aaa {**detail | error | verbose**}

Syntax Description

aaa	Enables debugging of IPv6 AAA events.
detail	Enables debugging of IPv6 multicast AAA details.
error	Enables debugging of IPv6 multicast AAA errors.
verbose	Enables debugging of IPv6 multicast AAA verbose.

Command Modes

Privileged EXEC(#)

Command History

Release	Modification
15.3(1)S	This command was introduced.

Usage Guidelines

You must configure multicast routing in an IPv6 environment. Use the **ipv6 multicast-routing** command in global configuration mode to enable IPv6 multicast routing. The **ipv6 multicast-routing** command applies on all IPv6-enabled interfaces on a device, which are then automatically enabled for Protocol-Independent Multicast version 6 (PIMv6). PIM is used between devices so that the devices can track which multicast packets to forward to each other and to the devices that are on the directly connected LANs.

Examples

The following example shows how to enable debugging of IPv6 multicast AAA information:

```
Device# debug ipv6 multicast aaa detail
AAA details debugging is on
Device# debug ipv6 multicast aaa error
AAA errors debugging is on
Device# debug ipv6 multicast aaa verbose
AAA verbose debugging is on
```

Related Commands

Command	Description
ipv6 multicast-routing	Enables multicast routing using MLD on all IPv6-enabled interfaces of the device and enables multicast forwarding.

debug ipv6 multicast rpf

To enable debugging of Reverse Path Forwarding (RPF) events related to IPv6 multicast routing, use the **debug ipv6 multicast rpf** command in privileged EXEC mode. To disable debugging of events, use the **no** form of this command.

debug ipv6 multicast rpf

no debug ipv6 multicast rpf

Syntax Description

rpf	Enables debugging of IPv6 multicast RPF events.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.3(1)S	This command was introduced.

Usage Guidelines

You must configure multicast routing in an IPv6 environment. Use the **ipv6 multicast-routing** command in global configuration mode to enable IPv6 multicast routing. The **ipv6 multicast-routing** command applies on all IPv6-enabled interfaces on a device, which are then automatically enabled for Protocol-Independent Multicast version 6 (PIMv6). PIM is used between devices so that the devices can track which multicast packets to forward to each other and to the devices that are on the directly connected LANs.

Examples

The following example shows how to enable debugging of IPv6 multicast RPF events:

```
Device# debug ipv6 multicast rpf
IPv6 Multicast RPF debugging is on
```

Related Commands

Command	Description
ipv6 multicast-routing	Enables multicast routing using MLD on all IPv6-enabled interfaces of the device and enables multicast forwarding.

debug ipv6 multicast rwatch

To enable debugging of route watch tracking events related to IPv6 multicast routing, use the **debug ipv6 multicast rwatch** command in privileged EXEC mode. To disable debugging of events, use the **no** form of this command.

debug ipv6 multicast rwatch

no debug ipv6 multicast rwatch

Syntax Description

rwatch	Enables debugging of IPv6 multicast route watch tracking events.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.3(1)S	This command was introduced.

Usage Guidelines

You must configure multicast routing in an IPv6 environment. Use the **ipv6 multicast-routing** command in global configuration mode to enable IPv6 multicast routing. The **ipv6 multicast-routing** command applies on all IPv6-enabled interfaces on a device, which are then automatically enabled for Protocol-Independent Multicast version 6 (PIMv6). PIM is used between devices so that the devices can track which multicast packets to forward to each other and to the devices that are on the directly connected LANs.

Examples

The following example shows how to enable debugging of IPv6 multicast route watch tracking events:

```
Device# debug ipv6 multicast rwatch
IPv6 Route-watch debugging is on
```

Related Commands

Command	Description
ipv6 multicast-routing	Enables multicast routing using MLD on all IPv6-enabled interfaces of the device and enables multicast forwarding.

debug ipv6 nat

To display debug messages for Network Address Translation--Protocol Translation (NAT-PT) translation events, use the **debug ipv6 nat**command in privileged EXEC mode. To disable debug messages for NAT-PT translation events, use the **no** form of this command.

debug ipv6 nat [**detailed**| **port**]

no debug ipv6 nat [**detailed**| **port**]

Syntax Description

detailed	(Optional) Displays detailed information about NAT-PT translation events.
port	(Optional) Displays port allocation events.

Command Default

Debugging for NAT-PT translation events is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(13)T	This command was introduced.
12.3(2)T	The **port** keyword was added to support Port Address Translation (PAT), or overload, multiplexing multiple IPv6 addresses to a single IPv4 address or to an IPv4 address pool.

Usage Guidelines

The **debug ipv6 nat**command can be used to troubleshoot NAT-PT translation issues. If no keywords are specified, debugging messages for all NAT-PT protocol translation events are displayed.

Note By default, the network server sends the output from **debug** commands and system error messages to the console. To redirect debugging output, use the logging command options within global configuration mode. Destinations are the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server.

Caution Because the **debug ipv6 nat** command generates a substantial amount of output, use it only when traffic on the IPv6 network is low, so other activity on the system is not adversely affected.

Examples

The following example shows output for the **debug ipv6 nat**command:

```
Router# debug ipv6 nat
00:06:06: IPv6 NAT: icmp src (3002::8) -> (192.168.124.8), dst (2001::2) -> (192.168.123.2)
00:06:06: IPv6 NAT: icmp src (192.168.123.2) -> (2001::2), dst (192.168.124.8) -> (3002::8)
00:06:06: IPv6 NAT: icmp src (3002::8) -> (192.168.124.8), dst (2001::2) -> (192.168.123.2)
00:06:06: IPv6 NAT: icmp src (192.168.123.2) -> (2001::2), dst (192.168.124.8) -> (3002::8)
00:06:06: IPv6 NAT: tcp src (3002::8) -> (192.168.124.8), dst (2001::2) -> (192.168.123.2)
00:06:06: IPv6 NAT: tcp src (192.168.123.2) -> (2001::2), dst (192.168.124.8) -> (3002::8)
00:06:06: IPv6 NAT: tcp src (3002::8) -> (192.168.124.8), dst (2001::2) -> (192.168.123.2)
00:06:06: IPv6 NAT: tcp src (3002::8) -> (192.168.124.8), dst (2001::2) -> (192.168.123.2)
00:06:06: IPv6 NAT: tcp src (3002::8) -> (192.168.124.8), dst (2001::2) -> (192.168.123.2)
00:06:06: IPv6 NAT: tcp src (192.168.123.2) -> (2001::2), dst (192.168.124.8) -> (3002::8)
```

The table below describes the significant fields shown in the display.

Table 1: debug ipv6 nat Field Descriptions

Field	Description
IPv6 NAT:	Indicates that this is a NAT-PT packet.
icmp	Protocol of the packet being translated.
src (3000::8) -> (192.168.124.8)	The source IPv6 address and the NAT-PT mapped IPv4 address. **Note** If mapping IPv4 hosts to IPv6 hosts the first address would be an IPv4 address, and the second address an IPv6 address.
dst (2001::2) -> (192.168.123.2)	The destination IPv6 address and the NAT-PT mapped IPv4 address. **Note** If mapping IPv4 hosts to IPv6 hosts the first address would be an IPv4 address, and the second address an IPv6 address.

The following example shows output for the **debug ipv6 nat**command with the **detailed** keyword:

```
Router# debug ipv6 nat detailed
00:14:12: IPv6 NAT: address allocated 192.168.124.8
00:14:16: IPv6 NAT: deleted a NAT entry after timeout
```

debug ipv6 nd

To display debug messages for IPv6 Internet Control Message Protocol (ICMP) neighbor discovery transactions, use the **debug ipv6 nd**command in privileged EXEC mode. To disable debug messages for IPv6 ICMP neighbor discovery transactions, use the **no** form of this command.

debug ipv6 nd

no debug ipv6 nd

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging for IPv6 ICMP neighbor discovery is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(2)T	This command was introduced.
12.2(4)T	The DAD: <*nnnn* ::*nn* :> is unique, DAD: duplicate link-local <*nnnn* ::*nn* :> on <*interface type* >, interface stalled, and Received NA for <*nnnn* ::*nn* :> on <*interface type* > from <*nnnn* ::*nn* :> fields were added to the command output.
12.0(21)ST	This command was integrated into Cisco IOS Release 12.0(21)ST.
12.0(22)S	This command was integrated into Cisco IOS Release 12.0(22)S.
12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(25)SG	This command was integrated into Cisco IOS Release 12.2(25)SG.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.

Usage Guidelines

This command can help determine whether the router is sending or receiving IPv6 ICMP neighbor discovery messages.

Note By default, the network server sends the output from debug commands and system error messages to the console. To redirect debug output, use the logging command options within global configuration mode. Destinations include the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server. For complete information on debug commands and redirecting debug output, refer to the *Cisco IOS Debug Command Reference* .

Examples

The following example shows output for the **debug ipv6 nd**command:

```
Router# debug ipv6 nd
13:22:40:ICMPv6-ND:STALE -> DELAY:2000:0:0:3::2
13:22:45:ICMPv6-ND:DELAY -> PROBE:2000:0:0:3::2
13:22:45:ICMPv6-ND:Sending NS for 2000:0:0:3::2 on FastEthernet0/0
13:22:45:ICMPv6-ND:Received NA for 2000:0:0:3::2 on FastEthernet0/0 from 2000:0:0:3::2
13:22:45:ICMPv6-ND:PROBE -> REACH:2000:0:0:3::2
13:22:45:ICMPv6-ND:Received NS for 2000:0:0:3::1 on FastEthernet0/0 from
FE80::203:A0FF:FED6:1400
13:22:45:ICMPv6-ND:Sending NA for 2000:0:0:3::1 on FastEthernet0/0
13:23:15: ICMPv6-ND: Sending NS for FE80::1 on Ethernet0/1
13:23:16: ICMPv6-ND: DAD: FE80::1 is unique.
13:23:16: ICMPv6-ND: Sending NS for 2000::2 on Ethernet0/1
13:23:16: ICMPv6-ND: Sending NS for 3000::3 on Ethernet0/1
13:23:16: ICMPv6-ND: Sending NA for FE80::1 on Ethernet0/1
13:23:17: ICMPv6-ND: DAD: 2000::2 is unique.
13:23:53: ICMPv6-ND: Sending NA for 2000::2 on Ethernet0/1
13:23:53: ICMPv6-ND: DAD: 3000::3 is unique.
13:23:53: ICMPv6-ND: Sending NA for 3000::3 on Ethernet0/1
3d19h: ICMPv6-ND: Sending NS for FE80::2 on Ethernet0/2
3d19h: ICMPv6-ND: Received NA for FE80::2 on Ethernet0/2 from FE80::2
3d19h: ICMPv6-ND: DAD: duplicate link-local FE80::2 on Ethernet0/2,interface stalled
3d19h: %IPV6-4-DUPLICATE: Duplicate address FE80::2 on Ethernet0/2
3d19h: ICMPv6-ND: Sending NS for 3000::4 on Ethernet0/3
3d19h: ICMPv6-ND: Received NA for 3000::4 on Ethernet0/3 from 3000::4
3d19h: %IPV6-4-DUPLICATE: Duplicate address 3000::4 on Ethernet0/3
```

The table below describes the significant fields shown in the display.

Table 2: debug ipv6 nd Field Descriptions

Field	Description
13:22:40:	Indicates the time (hours:minutes:seconds) at which the ICMP neighbor discovery event occrred.
ICMPv6-ND	Indicates that a state change is occurring for an entry in the IPv6 neighbors cache.
STALE	Stale state. This state of an neighbor discovery cache entry used to be "reachable," but is now is "stale" due to the entry not being used. In order to use this address, the router must go through the neighbor discovery process in order to confirm reachability.

Field	Description
DELAY	Delayed state. Reachability for this ND cache entry is currently being reconfirmed. While in the delay state, upper-layer protocols may inform IPv6 that they have confirmed reachability to the entry. Therefore, there is no need to send a neighbor solicitation for the entry.
PROBE	Probe state. While in the probe state, if no confirmation is received from the upper-layer protocols about the reachability of the entry, a neighbor solicitation message is sent. The entry remains in the "probe" state until a neighbor advertisement message is received in response to the neighbor solicitation message.
Sending NS for...	Sending a neighbor solicitation message. In the example output, a neighbor solicitation message is sent on Fast Ethernet interface 0/0 to determine the link-layer address of 2000:0:0:3::2 on Fast Ethernet interface 0/0.
Received NA for...	Received a neighbor advertisement message. In the example output, a neighbor advertisement message is received from the address 2000:0:0:3::2 (the second address) that includes the link-layer address of 2000:0:0:3::2 (first address) from Ethernet interface 0/0.
REACH	Reachable state. An ND cache entry in this state is considered reachable, and the corresponding link-layer address can be used without needing to perform neighbor discovery on the address.
Received NS for...	Received neighbor solicitations. In the example output, the address FE80::203:A0FF:FED6:1400 (on Fast Ethernet interface 0/0) is trying to determine the link-local address of 2000:0:0:3::1.
Sending NA for...	Sending for neighbor advertisements. In the example output, a neighbor advertisement containing the link-layer address of 2000:0:0:3::1 (an address assigned to the Fast Ethernet interface 0/0 address) was sent.
DAD: FE80::1 is unique.	Duplicate address detection processing was performed on the unicast IPv6 address (a neighbor solicitation message was not received in response to a neighbor advertisement message that contained the unicast IPv6 address) and the address is unique.

Field	Description
3d19h:	Indicates time (days, hours) since the last reboot of the event occurring; 3d19h: indicates the time (since the last reboot) of the event occurring was 3 days and 19 hours ago.
DAD: duplicate link-local FE80::2 on Ethernet0/2, interface stalled	Duplicate address detection processing was performed on the link-local IPv6 address (the link-local address FE80::2 is used in the example). A neighbor advertisement message was received in response to a neighbor solicitation message that contained the link-local IPv6 address. The address is not unique, and the processing of IPv6 packets is disabled on the interface.
%IPV6-4-DUPLICATE: Duplicate address...	System error message indicating the duplicate address.
Received NA for 3000::4 on Ethernet0/3 from 3000::4	Duplicate address detection processing was performed on the global IPv6 address (the global address 3000::4 is used in the example). A neighbor advertisement message was received in response to a neighbor solicitation message that contained the global IPv6 address. The address is not unique and is not used.

Related Commands

Command	Description
debug ipv6 icmp	Displays debug messages for IPv6 ICMP transactions.
show ipv6 neighbors	Displays IPv6 neighbor discovery cache information.

debug ipv6 ospf

To display debugging information for Open Shortest Path First (OSPF) for IPv6, use the **debug ipv6 ospf** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipv6 ospf [**adj**| **ipsec**| **database-timer**| **flood**| **hello**| **lsa-generation**| **retransmission**]

no debug ipv6 ospf [**adj**| **ipsec**| **database-timer**| **flood**| **hello**| **lsa-generation**| **retransmission**]

Syntax Description

adj	(Optional) Displays adjacency information.
ipsec	(Optional) Displays the interaction between OSPF and IPSec in IPv6 networks, including creation and removal of policy definitions.
database-timer	(Optional) Displays database-timer information.
flood	(Optional) Displays flooding information.
hello	(Optional) Displays hello packet information.
l2api	(Optional) Enables layer 2 and layer 3 application program interface (API) debugging.
lsa-generation	(Optional) Displays link-state advertisement (LSA) generation information for all LSA types.
retransmission	(Optional) Displays retransmission information.

Command Default

Debugging of OSPF for IPv6 is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(24)S	This command was introduced.
12.2(15)T	This command was integrated in Cisco IOS Release 12.2(15)T.
12.2(18)S	This command was integrated in Cisco IOS Release 12.2(18)S.
12.3(4)T	The **ipsec** keyword was added to support OSPF for IPv6 authentication for IPSec.

Release	Modification
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
12.4(25)T	The **l2api** keyword was added.

Usage Guidelines

Consult Cisco technical support before using this command.

Examples

The following example displays adjacency information for OSPF for IPv6:

```
Router# debug ipv6 ospf adj
```

debug ipv6 ospf database-timer rate-limit

To display debugging information about the current wait-time used for SPF scheduling, use the **debug ipv6 ospf database-timer rate-limit**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipv6 ospf database-timer rate-limit [*acl-number*]

no debug ipv6 ospf database-timer rate-limit

Syntax Description

acl-number	(Optional) Access list number.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SRC	This command was introduced.
12.2(33)SB	This command was integrated into Cisco IOS Release 12.2(33)SB.

Usage Guidelines

Consult Cisco technical support before using this command.

Examples

The following example shows how to turn on debugging for SPF scheduling:

```
Router# debug ipv6 ospf database-timer rate-limit
```

debug ipv6 ospf events

To display information on Open Shortest Path First (OSPF)-related events, such as designated router selection and shortest path first (SPF) calculation, use the **debug ipv6 ospf events** command in privileged EXEC com mand. To disable debugging output, use the **no** form of this command.

debug ipv6 ospf events

no debug ipv6 ospf events

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(24)S	This command was introduced.
12.2(15)T	This command was integrated into Cisco IOS Release 12.2(15)T.
12.2(18)S	This command was integrated into Cisco IOS Release 12.2(18)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.

Command History

12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
12.2(33)XNE	This command was modified. It was integrated into Cisco IOS Release 12.2(33)XNE.

Usage Guidelines

Consult Cisco technical support before using this command.

Examples

The following example displays information on OSPF-related events:

```
Router#
debug ipv6 ospf events
```

debug ipv6 ospf graceful-restart

To enable debugging for IPv6 graceful-restart-related events, use the **debug ipv6 ospf graceful-restart**command in privileged EXEC mode.

debug ipv6 ospf graceful-restart

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 2.1	This command was introduced.
15.0(1)M	This command was integrated into Cisco IOS Release 15.0(1)M.
12.2(33)SRE	This command was modified. It was integrated into Cisco IOS Release 12.2(33)SRE.

Usage Guidelines

The **debug ipv6 ospf graceful-restart** command helps troubleshoot graceful-restart-related events on both graceful-restart-capable and graceful-restart-aware routers.

Examples

The following example enables debugging for graceful-restart-related events:

```
Router# debug ipv6 ospf graceful-restart
00:03:41: OSPFv3: GR timer started for ospf process 1 for 120 secs,
00:03:43: OSPFv3: GR  Build Grace LSA for interface Ethernet0/0
00:03:43: OSPFv3: GR Flood grace lsa on Ethernet0/0
00:03:43: OSPFv3: GR complete check for area 0 process 1
00:03:43: OSPFv3: GR wait, Ethernet0/0 in area 0 not yet complete
00:03:45: OSPFv3: GR Re-flood Grace LSA on Ethernet0/0
00:04:01: OSPFv3: GR  initial wait expired
00:04:01: OSPFv3: GR complete check for area 0 process 1
00:04:01: OSPFv3: GR wait, Ethernet0/0 in area 0 not yet complete
00:04:07: OSPFv3: GR complete check for area 0 process 1
00:04:07: OSPFv3: GR re-sync completed in area 0, process 1
00:04:07: OSPFv3: GR complete check for process 1
00:04:07: OSPFv3: process 1: GR re-sync completed for all neighbors
00:04:07: OSPFv3: scheduling rtr lsa for area 0 process 1
00:04:07: OSPFv3: Post GR, flood maxaged grace-LSA on Ethernet0/0
```

Related Commands

Command	Description
graceful-restart	Enables the OSPFv3 graceful restart feature on a graceful-restart-capable router.
graceful-restart helper	Enables the OSPFv3 graceful restart feature on a graceful-restart-aware router.
show ipv6 ospf graceful-restart	Displays OSPFv3 graceful restart information.

debug ipv6 ospf lsdb

To display database modifications for Open Shortest Path First (OSPF) for IPv6, use the **d ebug ipv6 ospf lsdb**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipv6 ospf lsdb

no debug ipv6 ospf lsdb

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(24)S	This command was introduced.
12.2(15)T	This command was integrated into Cisco IOS Release 12.2(15)T.
12.2(18)S	This command was integrated into Cisco IOS Release 12.2(18)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.

Usage Guidelines

Consult Cisco technical support before using this command.

Examples

The following example displays database modification information for OSPF for IPv6:

```
Router# debug ipv6 ospf lsdb
```

debug ipv6 ospf monitor

To display debugging information about the current wait-time used for shortest path first (SPF) scheduling, use the **debug ipv6 ospf monitor** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipv6 ospf monitor

no debug ipv6 ospf monitor

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SRC	This command was introduced.

Command History

12.2(33)SB	This command was integrated into Cisco IOS Release 12.2(33)SB.

Usage Guidelines

Consult Cisco technical support before using this command.

Examples

The following example shows debugging information about SPF scheduling:

```
Router# debug ipv6 ospf monitor
Sep 27 08:29:49.319: OSPFv3: Schedule SPF in area 0
        Change in LS ID 0.0.0.0, LSA type P
*Sep 27 08:29:49.327: OSPFv3: reset throttling to 5000ms next wait-interval 10000ms
*Sep 27 08:29:49.327: OSPFv3: schedule SPF: spf_time 00:09:36.032 wait_interval 5000ms
IOU_Topvar#
*Sep 27 08:29:54.331: OSPFv3: Begin SPF at 581.036ms, process time 40ms
*Sep 27 08:29:54.331:       spf_time 00:09:36.032, wait_interval 5000ms
*Sep 27 08:29:54.331: OSPFv3: Setting next wait-interval to 10000ms
*Sep 27 08:29:54.331: OSPFv3: End SPF at 581.036ms, Total elapsed time 0ms
*Sep 27 08:29:54.331:       Schedule time 00:09:41.036, Next wait_interval 10000ms
*Sep 27 08:29:54.331:       Intra: 0ms, Inter: 0ms, External: 0ms
*Sep 27 08:29:54.331:       R: 0, N: 0
*Sep 27 08:29:54.331:       SN: 0, SA: 0, X5: 0, X7: 0
*Sep 27 08:29:54.331:       SPF suspends: 0 intra, 0 total
```

debug ipv6 ospf packet

To display information about each Open Shortest Path First (OSPF) for IPv6 packet received, use the **debug ipv6 ospf packet** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipv6 ospf packet

no debug ipv6 ospf packet

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(24)S	This command was introduced.
12.2(15)T	This command was integrated into Cisco IOS Release 12.2(15)T.
12.2(18)S	This command was integrated into Cisco IOS Release 12.2(18)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.

Usage Guidelines

Consult Cisco technical support before using this command.

Examples

The following example displays information about each OSPF for IPv6 packet received:

```
Router# debug ipv6 ospf packet
```

debug ipv6 ospf spf statistic

To display statistical information while running the shortest path first (SPF) algorithm, use the **debug ipv6 ospf spf statistic**command in privileged EXEC mode. To disable the debugging output, use the **no** form of this command.

debug ipv6 ospf spf statistic

no debug ipv6 ospf spf statistic

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(24)S	This command was introduced.
12.2(15)T	This command was integrated into Cisco IOS Release 12.2(15)T.
12.2(18)S	This command was integrated into Cisco IOS Release 12.2(18)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.

Usage Guidelines

The **debug ipv6 ospf spf statistic** command displays the SPF calculation times in milliseconds, the node count, and a time stamp. Consult Cisco technical support before using this command.

Examples

The following example displays statistical information while running the SPF algorithm:

```
Router# debug ipv6 ospf spf statistics
```

Related Commands

Command	Description
debug ipv6 ospf	Displays debugging information for the OSPFv3 for IPv6 feature.
debug ipv6 ospf events	Displays information on OSPFv3-related events.

Command	Description
debug ipv6 ospf packet	Displays information about each OSPFv3 packet received.

debug ipv6 packet

To display debug messages for IPv6 packets, use the **debug ipv6 packet**command in privileged EXEC mode. To disable debug messages for IPv6 packets, use the **no** form of this command.

debug ipv6 packet [**access-list** *access-list-name*] **[detail]**

no debug ipv6 packet [**access-list** *access-list-name*] **[detail]**

Syntax Description

access-list *access-list-name*	(Optional) Specifies an IPv6 access list. The access list name cannot contain a space or quotation mark, or begin with a numeric
detail	(Optional) May display additional detailed information about the IPv6 packet.

Command Default

Debugging for IPv6 packets is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(2)T	This command was introduced.
12.0(21)ST	This command was integrated into Cisco IOS Release 12.0(21)ST.
12.0(22)S	This command was integrated into Cisco IOS Release 12.0(22)S.
12.0(23)S	The **access-list** and **detail** keywords, and the *access-list-name*argument, were added.
12.2(13)T	The **access-list** and **detail** keywords, and the *access-list-name*argument, were added.
12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(25)SG	This command was integrated into Cisco IOS Release 12.2(25)SG.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.

Usage Guidelines

The **debug ipv6 packet**command is similar to the **debug ip packet**command, except that it is IPv6-specific.

Note By default, the network server sends the output from debug commands and system error messages to the console. To redirect debug output, use the logging command options within global configuration mode. Destinations include the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server. For complete information on debug commands and redirecting debug output, refer to the *Cisco IOS Debug Command Reference* .

IPv6 debugging information includes packets received, generated, and forwarded. Fast-switched packets do not generate messages. When an IPv6 access list is specified by using the **access-list**keyword and *access-list-name*argument, only packets matching the access list permit entries are displayed.

Caution Because the **debug ipv6 packet** command generates a substantial amount of output, use it only when traffic on the IPv6 network is low, so other activity on the system is not adversely affected.

Examples

The following example shows output for the **debug ipv6 packet**command:

```
Router# debug ipv6 packet
13:25:40:IPV6:source 2000:0:0:3::1 (local)
13:25:40:      dest 2000:0:0:3::2 (FastEthernet0/0)
13:25:40:      traffic class 96, flow 0x0, len 143+195, prot 6, hops 64, originating
13:25:40:IPv6:Sending on FastEthernet0/0
13:25:40:IPV6:source 2000:0:0:3::2 (FastEthernet0/0)
13:25:40:      dest 2000:0:0:3::1
13:25:40:      traffic class 96, flow 0x0, len 60+14, prot 6, hops 64, forward to ulp
13:25:45:IPV6:source FE80::203:E4FF:FE12:CC1D (local)
13:25:45:      dest FF02::9 (Ethernet1/1)
13:25:45:      traffic class 112, flow 0x0, len 72+1428, prot 17, hops 255, originating
13:25:45:IPv6:Sending on Ethernet1/1
13:25:45:IPV6:source FE80::203:E4FF:FE12:CC00 (local)
13:25:45:      dest 2000:0:0:3::2 (FastEthernet0/0)
13:25:45:      traffic class 112, flow 0x0, len 72+8, prot 58, hops 255, originating
13:25:45:IPv6:Sending on FastEthernet0/0
13:25:45:IPV6:source 2000:0:0:3::2 (FastEthernet0/0)
13:25:45:      dest FE80::203:E4FF:FE12:CC00
13:25:45:      traffic class 112, flow 0x0, len 64+14, prot 58, hops 255, forward to ulp
13:25:45:IPV6:source FE80::203:A0FF:FED6:1400 (FastEthernet0/0)
13:25:45:      dest 2000:0:0:3::1
13:25:45:      traffic class 112, flow 0x0, len 72+14, prot 58, hops 255, forward to ulp
```

The table below describes the significant fields shown in the display.

Table 3: debug ipv6 packet Field Descriptions

Field	Description
IPV6:	Indicates that this is an IPv6 packet.
source 2000:0:0:3::1 (local)	The source address in the IPv6 header of the packet.
dest 2000:0:0:3::2 (FastEthernet0/0)	The destination address in the IPv6 header of the packet.

Field	Description
traffic class 96	The contents of the traffic class field in the IPv6 header.
flow 0x0	The contents of the flow field of the IPv6 header. The flow field is used to label sequences of packets for which special handling is necessary by IPv6 routers.
len 64+14	The length of the IPv6 packet. The length is expressed as two numbers with a plus (+) character between the numbers. The first number is the length of the IPv6 portion (IPv6 header length plus payload length). The second number is the entire datagram size minus the first number.
prot 6	The protocol field in the IPv6 header. Describes the next layer protocol that is carried by the IPv6 packet. In the example, the protocol 58 signifies that the next layer protocol is ICMPv6.
hops 64	The hops field in the IPv6 packet. This field is similar in function to the IPv4 time-to-live field.
originating	The presence of this field indicates that the packet shown was originated by the router.
Sending on FastEthernet0/0	Specifies the interface on which the packet was sent.
forward to ulp	Indicates that the packet was received by the router at the destination address and was forwarded to an upper-layer protocol (ulp) for processing.

debug ipv6 pim

To enable debugging on Protocol Independent Multicast (PIM) protocol activity, use the **debug ipv6 pim**command in privileged EXEC mode. To restore the default value, use the **no** form of this command.

debug ipv6 pim [*group-name*| *group-address*| **interface** *interface-type*| **bsr**| **group**| **neighbor**]

no debug ipv6 pim [*group-name*| *group-address*| **interface** *interface-type*| **bsr**| **group**| **neighbor**]

Syntax Description

group-name | *group-address*	(Optional) IPv6 address or name of the multicast group.
interface *interface-type*	(Optional) Displays debugging statistics about a specific interface type.
bsr	(Optional) Displays debugging statistics specific to bootstrap router (BSR) protocol operation.
group	(Optional) Displays debugging information about group-related activity.
neighbor	(Optional) Displays debugging statistics related to hello message processing and neighbor cache management.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(2)T	This command was introduced.
12.2(18)S	This command was integrated into Cisco IOS Release 12.2(18)S.
12.0(26)S	This command was integrated into Cisco IOS Release 12.0(26)S.
12.0(28)S	The **bsr** keyword was added.
12.2(25)S	The **bsr** keyword was added.
12.3(11)T	The **bsr** keyword was added.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(25)SG	This command was integrated into Cisco IOS Release 12.2(25)SG.

Release	Modification
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
Cisco IOS XE Release 2.1	This command was introduced on Cisco ASR 1000 Series Routers.

Usage Guidelines

This command helps discover whether the PIM protocol activities are working correctly.

The messages displayed by the **debug ipv6 pim**command show all PIM protocol messages, such as joins and prunes, received from or sent to other routers. Use this command in conjunction with **debug ipv6 mld**to display additional multicast activity, to learn more information about the multicast routing process, or to learn why packets are forwarded out of particular interfaces.

Examples

The following example enables debugging on PIM activity:

```
Router# debug ipv6 pim
```

Related Commands

Command	Description
debug ipv6 mld	Enables debugging on MLD protocol activity.

debug ipv6 pim df-election

To display debug messages for Protocol Independent Multicast (PIM) bidirectional designated forwarder (DF) election message processing, use the **debug ipv6 pim df-election**command in privileged EXEC mode. To disable debug messages for PIM bidirectional DF election message processing, use the **no** form of this command.

debug ipv6 pim df-election [**interface** *type number*] [**rp** *rp-name*| *rp-address*]

no debug ipv6 pim df-election [**interface** *type number*] [**rp** *rp-name*| *rp-address*]

Syntax Description

interface	(Optional) Specifies that debug messages on a specified interface will be displayed.
type number	(Optional) Interface type and number. For more information, use the question mark (?) online help function.
rp	(Optional) Specifies that debug messages on a specified Route Processor (RP) will be displayed.
rp-name	(Optional) The name of the specified RP.
rp-address	(Optional) The IPv6 address of the specified RP.

Command Default

Debugging for PIM bidirectional DF election message processing is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(7)T	This command was introduced.
12.2(25)S	This command was integrated into Cisco IOS Release 12.2(25)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.

Usage Guidelines

Use the **debug ipv6 pim df-election**command if traffic is not flowing properly when operating in PIM bidirectional mode or if the **show ipv6 pim df** and **show ipv6 pim df winner**commandsdo not display the expected information.

Examples

The following example shows how to enable debugging for PIM bidirectional DF election message processing on Ethernet interface 1/0 and at 200::1:

```
Route# debug ipv6 pim df-election interface ethernet 1/0 rp 200::1
```

Related Commands

Command	Description
ipv6 pim rp-address	Configures the address of a PIM RP for a particular group range.
show ipv6 pim df	Displays the DF-election state of each interface for each RP.
show ipv6 pim df winner	Displays the DF-election winner on each interface for each RP.

debug ipv6 pim limit

To enable debugging for Protocol Independent Multicast (PIM) interface limits, use the **debug ipv6 pim limit** command in privileged EXEC mode. To restore the default value, use the **no** form of this command.

debug ipv6 pim limit [*group*]

no debug ipv6 pim limit

Syntax Description

group	(Optional) Specific group to be debugged.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SRE	This command was introduced.

Usage Guidelines

Use the **debug ipv6 pim limit** command to display debugging information for interface limits and costs. Use the optional *group* argument to specify a particular group to debug.

Examples

The following example enables PIM interface limit debugging:

```
Router# debug ipv6 pim limit
```

Related Commands

Command	Description
ipv6 multicast limit	Configures per-interface mroute state limiters in IPv6.
ipv6 multicast limit cost	Applies a cost to mroutes that match per interface mroute state limiters in IPv6.

debug ipv6 policy

To enable debugging of IPv6 policy routing packet activity, use the **debug ipv6 policy** command in user EXEC or privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug ipv6 policy [*access-list-name*]

no debug ipv6 policy [*access-list-name*]

Syntax Description

access-list-name	(Optional) Name of the IPv6 access list. Names cannot contain a space or quotation mark or begin with a numeric.

Command Default

If no access list is specified using the optional *access-list-name* argument, information about all policy-matched and policy-routed packets is displayed.

Command Modes

User EXEC (>)

Privileged EXEC (#)

Command History

Release	Modification
12.3(7)T	This command was introduced.
12.2(30)S	This command was integrated into Cisco IOS Release 12.2(30)S.
12.2(33)SXI4	This command was integrated into Cisco IOS Release 12.2(33)SXI4.
Cisco IOS XE Release 3.2S	This command was integrated into Cisco IOS XE Release 3.2S.
15.1(1)SY	This command was integrated into Cisco IOS Release 15.1(1)SY.

Usage Guidelines

After you configure IPv6 policy routing, use the **debug ipv6 policy** command to verify that IPv6 policy-based routing (PBR) is policy-routing packets normally. Policy routing analyzes various parts of the packet and then routes the packet based on certain user-defined attributes in the packet. The **debug ipv6 policy** command helps you determine what policy is followed during routing. It displays information about whether a packet matches the given criteria, and if yes, the resulting routing information for the packet.

Do not use the **debug ipv6 policy** command unless you suspect a problem with IPv6 PBR policy routing.

Examples

The following example shows how to enable debugging of IPv6 policy routing packet activity. The output of this command is self-explanatory:

```
Device# debug ipv6 policy

00:02:38:IPv6 PBR:Ethernet0/0, matched src 2003::90 dst 2001:DB8::1 protocol 58
00:02:38:IPv6 PBR:set nexthop 2001:DB8::F, interface Ethernet1/0
00:02:38:IPv6 PBR:policy route via Ethernet1/0/2001:DB8::F
```

debug ipv6 pool

To enable debugging on IPv6 prefix pools, use the debug ipv6 pool command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug ipv6 pool

no debug ipv6 pool

Syntax Description

This command has no keywords or arguments.

Command Default

No debugging is active.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(13)T	This command was introduced.

Examples

The following example enables debugging for IPv6 prefix pools:

```
Router# debug ipv6 pool
2w4d: IPv6 Pool: Deleting route/prefix 2001:0DB8::/29 to Virtual-Access1 for cisco
2w4d: IPv6 Pool: Returning cached entry 2001:0DB8::/29 for cisco on Virtual-Access1 to
pool1
2w4d: IPv6 Pool: Installed route/prefix 2001:0DB8::/29 to Virtual-Access1 for cisco
```

Related Commands

Command	Description
ipv6 local pool	Configures a local IPv6 prefix pool.
show ipv6 interface	Displays the usability status of interfaces configured for IPv6.
show ipv6 local pool	Displays information about defined IPv6 prefix pools.

debug ipv6 rip

To display debug messages for IPv6 Routing Information Protocol (RIP) transactions, use the **debug ipv6 rip** command in privileged EXEC mode. To disable debug messages for IPv6 RIP routing transactions, use the **no** form of this command.

Cisco IOS XE Release 3.9S, Cisco IOS Release 15.3(2)S, and Later Releases

debug ipv6 rip [*interface-type interface-number*] [**vrf** *vrf-name*]

no debug ipv6 rip [*interface-type interface-number*] [**vrf** *vrf-name*]

Releases Prior to Cisco IOS XE Release 3.9S and Cisco IOS Release 15.3(2)S

debug ipv6 rip [*interface-type interface-number*]

no debug ipv6 rip [*interface-type interface-number*]

Syntax Description

interface-type	(Optional) Interface type for which to display the debug messages.
interface-number	(Optional) Interface number for which to display the debug messages.
vrf *vrf-name*	(Optional) Displays information about the specified virtual routing and forwarding (VRF) instance.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(2)T	This command was introduced.
12.0(21)ST	This command was integrated into Cisco IOS Release 12.0(21)ST.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(25)SG	This command was integrated into Cisco IOS Release 12.2(25)SG.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
Cisco IOS XE Release 2.1	This command was implemented on Cisco 1000 Series Aggregation Services Routers.

Release	Modification
Cisco IOS XE Release 3.9S	This command was modified. The **vrf** *vrf-name* keyword-argument pair was added.
15.3(2)S	This command was integrated into Cisco IOS Release 15.3(2)S.
15.3(3)M	This command was integrated into Cisco IOS Release 15.3(3)M.

Usage Guidelines

The **debug ipv6 rip** command is similar to the **debug ip rip** command, except that it is IPv6-specific.

Note By default, the network server sends the output from debug commands and system error messages to the console. To redirect debug output, use the **logging** command in global configuration mode. Destinations include the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server. For complete information on debug commands and redirecting debug output, refer to the *Cisco IOS Debug Command Reference*.

Use the **debug ipv6 rip** command to enable IPv6 RIP debugging for RIP packets that are sent and received on all device interfaces. Use the **debug ipv6 rip** *interface-type interface-number* command to enable IPv6 RIP debugging for RIP packets that are sent and received only on the specified interface.

Use the **debug ipv6 rip vrf** *vrf-name* command to troubleshoot issues in the IPv6 RIP functionality when the VRF has already been enabled using a **vrf definition** *vrf-name* command. Ensure that the specified VRF name has already been defined. If a VRF name has not been defined, the following message is displayed:

```
% VRF <undefined VRF name> does not exist or does not have a RD.
```

Examples

The following is sample output from the **debug ipv6 rip** command:

```
Device# debug ipv6 rip

13:09:10:RIPng:Sending multicast update on Ethernet1/1 for as1_rip
13:09:10:       src=2001:DB8::1
13:09:10:       dst=2001:DB8:0:ABCD::1 (Ethernet1/1)
13:09:10:       sport=521, dport=521, length=32
13:09:10:       command=2, version=1, mbz=0, #rte=1
13:09:10:       tag=0, metric=1, prefix=::/0
13:09:28:RIPng:response received from 2001:DB8:0:0:E000::F on Ethernet1/1 for as1_rip
13:09:28:       src=FE80::202:FDFF:FE77:1E42 (Ethernet1/1)
13:09:28:       dst=FF02::9
13:09:28:       sport=521, dport=521, length=32
13:09:28:       command=2, version=1, mbz=0, #rte=1
13:09:28:       tag=0, metric=1, prefix=2000:0:0:1:1::/80
```

The above example shows two RIP packets; both are known as “responses” in RIP terminology and indicated by a “command” value of 2. The first is an update sent by the device, and the second is an update received by the device. Multicast update packets are sent to all neighboring IPv6 RIP devices (all devices that are on the same links as the device sending the update and have IPv6 RIP enabled). An IPv6 RIP device advertises the contents of its routing table to its neighbors by periodically sending update packets over those interfaces on which IPv6 RIP is configured. An IPv6 device may also send “triggered” updates immediately following a routing table change. In this case, the updates include only the changes to the routing table. An IPv6 RIP device may solicit the contents of the routing table of a neighboring device by sending a Request (command =1) message to the device. The device responds by sending an update (Response, command=2) containing

its routing table. In the example, the received response packet could be a periodic update from the address 2001:DB8:0:0:E000::F or a response to a RIP request message that was previously sent by the local device.

The following is sample output from the **debug ipv6 rip vrf** command:

```
Device# debug ipv6 rip vrf blue

RIP Routing Protocol debugging is on for vrf blue

Sending:
*Mar 15 11:23:08.508: RIPng: Sending multicast update on Ethernet0/0 for vrf for vrf blue
*Mar 15 11:23:08.508:        src=2001:DB8:0:1:FFFF:1234::5
*Mar 15 11:23:08.508:        dst=2001:DB8:0:1::1 (Ethernet0/0)
*Mar 15 11:23:08.508:        sport=521, dport=521, length=52
*Mar 15 11:23:08.508:        command=2, version=1, mbz=0, #rte=2
*Mar 15 11:23:08.508:        tag=0, metric=1, prefix=6000::/64
*Mar 15 11:23:08.508:        tag=0, metric=1, prefix=2000::/64
*Mar 15 11:23:08.508: RIPng: Packet waiting
*Mar 15 11:23:08.508: RIPng: Process vrf received own response on Loopback1

Receiving
*Mar 15 11:23:20.316: RIPng: Packet waiting
*Mar 15 11:23:20.316: RIPng: response received from FE80::A8BB:CCFF:FE00:7C00 on Ethernet0/0
 for vrf
*Mar 15 11:23:20.316:        src=2001:DB8:0:1:FFFF:1234::4 (Ethernet0/0)
*Mar 15 11:23:20.316:        dst=2001:DB8::1
*Mar 15 11:23:20.316:        sport=521, dport=521, length=32
*Mar 15 11:23:20.316:        command=2, version=1, mbz=0, #rte=1
*Mar 15 11:23:20.316:        tag=0, metric=1, prefix=AAAA::/64
```

The table below describes the significant fields shown in the display.

Table 4: debug ipv6 rip vrf Field Descriptions

Field	Description
src	The address from which the update was originated.
dst	The destination address for the update.
sport, dport, length	The source, destination ports and the length for the update. (IPv6 RIP uses port 521, as shown in the display.)
command	The command field within the RIP packet. A value of 2 indicates that the RIP packet is a response (update); a value of 1 indicates that the RIP packet is a request.
version	The version of IPv6 RIP being used. The current version is 1.
mbz	There must be a 0 (mbz) field within the RIP packet.
#rte	Indicates the number of routing table entries (RTEs) that the RIP packet contains.

Field	Description
tag metric prefix	The tag, metric, and prefix fields are specific to each RTE contained in the update. The tag field is intended to allow for the flagging of IPv6 RIP “internal” and “external” routes. The metric field is the distance metric from the device (sending this update) to the prefix. The prefix field is the IPv6 prefix of the destination being advertised.

Related Commands

Command	Description
clear ipv6 rip	Deletes routes from the IPv6 RIP routing table.
ipv6 rip vrf-mode enable	Enables VRF support for IPv6 RIP.
show ipv6 rip	Displays information about current IPv6 RIP processes.
vrf definition	Configures a VRF routing table instance.

debug ipv6 routing

To display debug messages for IPv6 routing table updates and route cache updates, use the **debug ipv6 routing**command in privileged EXEC mode. To disable debug messages for IPv6 routing table updates and route cache updates, use the **no** form of this command.

debug ipv6 routing

no debug ipv6 routing

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging for IPv6 routing table updates and route cache updates is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(2)T	This command was introduced.
12.0(21)ST	This command was integrated into Cisco IOS Release 12.0(21)ST.
12.0(22)S	This command was integrated into Cisco IOS Release 12.0(22)S.
12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(25)SG	This command was integrated into Cisco IOS Release 12.2(25)SG.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.

Usage Guidelines

The **debug ipv6 routing**command is similar to the **debug ip routing**command, except that it is IPv6-specific.

Note By default, the network server sends the output from debug commands and system error messages to the console. To redirect debug output, use the logging command options within global configuration mode. Destinations include the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server. For complete information on debug commands and redirecting debug output, refer to the *Cisco IOS Debug Command Reference* .

Examples

The following example shows output for the **debug ipv6 routing**command:

```
Router# debug ipv6 routing
13:18:43:IPv6RT0:Add 2000:0:0:1:1::/80 to table
13:18:43:IPv6RT0:Better next-hop for 2000:0:0:1:1::/80, [120/2]
13:19:09:IPv6RT0:Add 2000:0:0:2::/64 to table
13:19:09:IPv6RT0:Better next-hop for 2000:0:0:2::/64, [20/1]
13:19:09:IPv6RT0:Add 2000:0:0:2:1::/80 to table
13:19:09:IPv6RT0:Better next-hop for 2000:0:0:2:1::/80, [20/1]
13:19:09:IPv6RT0:Add 2000:0:0:4::/64 to table
13:19:09:IPv6RT0:Better next-hop for 2000:0:0:4::/64, [20/1]
13:19:37:IPv6RT0:Add 2000:0:0:6::/64 to table
13:19:37:IPv6RT0:Better next-hop for 2000:0:0:6::/64, [20/2]
```

The **debug ipv6 routing** command displays messages whenever the routing table changes. For example, the following message indicates that a route to the prefix 2000:0:0:1:1::/80 was added to the routing table at the time specified in the message.

```
13:18:43:IPv6RT0:Add 2000:0:0:1:1::/80 to table
```

The following message indicates that the prefix 2000:0:0:2::/64 was already in the routing table; however, a received advertisement provided a lower cost path to the prefix. Therefore, the routing table was updated with the lower cost path. (The [20/1] in the example is the administrative distance [20] and metric [1] of the better path.)

```
13:19:09:IPv6RT0:Better next-hop for 2000:0:0:2::/64, [20/1]
```

Related Commands

Command	Description
debug ipv6 rip	Displays debug messages for IPv6 RIP routing transactions.

debug ipv6 snooping

To enable debugging for security snooping information in IPv6, use the **debug ipv6 snooping**command in privileged EXEC mode.

debug ipv6 snooping [**binding-table**| **classifier**| **errors**| **feature-manager**| **filter** *acl*| **ha**| **hw-api**| **interface** *interface*| **memory**| **ndp-inspection**| **policy**| **vlan** *vlanid*| **switcher**| **filter** *acl*| **interface** *interface*| *vlan-id*]

no debug ipv6 snooping

Syntax Description

binding-table	(Optional) Displays information about the neighbor binding table.
classifier	(Optional) Displays information about the classifier.
errors	(Optional) Displays information about snooping security errors.
feature-manager	(Optional) Displays feature manager information.
filter *acl*	(Optional) Allows users to configure an access list to filter debugged traffic.
ha	(Optional) Displays information about high availability (HA) and stateful switchover (SSO).
hw-api	(Optional) Displays information about the hardware API.
interface *interface*	(Optional) Provides debugging information on a specified interface.
memory	(Optional) Displays information about security snooping memory.
ndp-inspection	(Optional) Displays information about Neighbor Discovery inspection.
policy	(Optional)
switcher	(Optional) Displays packets handled by the switcher.
vlan-id	(Optional) Provides debugging information about a specified VLAN ID.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(50)SY	This command was introduced.

Usage Guidelines

The **debug ipv6 snooping** command provides debugging output for IPv6 snooping information.

Because debugging output is assigned high priority in the CPU process, you should use debug commands only to troubleshoot specific problems or during troubleshooting sessions with Cisco technical support staff.

Examples

The following example enables debugging for all IPv6 snooping information:

```
Router# debug ipv6 snooping
```

debug ipv6 snooping raguard

To enable debugging for security snooping information in the IPv6 router advertisement (RA) guard feature, use the **debug ipv6 snooping raguard** command in privileged EXEC mode.

debug ipv6 snooping raguard [*filter*| *interface*| *vlanid*]

no debug ipv6 snooping raguard

Syntax Description

filter	(Optional) Allows users to configure an access list to filter debugged traffic.
interface	(Optional) Provides debugging information about a specified interface configured with the IPv6 RA guard feature.
vlanid	(Optional) Provides debugging information about a specified VLAN ID configured with the IPv6 RA guard feature.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(54)SG	This command was introduced.
12.2(50)SY	This command was integrated into Cisco IOS Release 12.2(50)SY.
15.2(4)S	This command was integrated into Cisco IOS Release 15.2(4)S.

Usage Guidelines

The **debug ipv6 snooping raguard** command provides debugging output for IPv6 RA guard events and errors that may occur.

Because debugging output is assigned high priority in the CPU process, you should use debug commands only to troubleshoot specific problems or during troubleshooting sessions with Cisco technical support staff. Also, you should use debug commands during periods of lower network traffic and fewer users. Debugging during these periods decreases the likelihood that increased debug command processing overhead will affect system use.

Examples

The following example shows the command enabling debugging for the IPv6 RA guard feature:

```
Router# debug ipv6 snooping raguard
```

Related Commands

Command	Description
ipv6 nd raguard	Applies the IPv6 RA guard feature.

debug ipv6 spd

To enable debugging output for the most recent Selective Packet Discard (SPD) state transition, use the **debug ipv6 spd**command in privileged EXEC mode.

debug ipv6 spd

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.1(3)T	This command was introduced.

Usage Guidelines

The **debug ipv6 spd** command enables debugging information to be reviewed for the most recent SPD state transition and any trend historical data.

Examples

The following example shows how to enable debugging for the most recent SPD state transition:

```
Router# debug ipv6 spd
```

debug ipv6 static

To enable Bidirectional Forwarding Detection for IPv6 (BFDv6) debugging, use the **debug ipv6 static**command in privileged EXEC mode.

debug ipv6 static

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 2.1.0	This command was introduced.
15.1(2)T	This command was modified. It was integrated into Cisco IOS Release 15.1(2)T.
15.1(1)SG	This command was integrated into Cisco IOS Release 15.1(1)SG.
15.1(1)SY	This command was modified. Support for IPv6 was added to Cisco IOS Release 15.1(1)SY.

Usage Guidelines

Use the **debug ipv6 static**command to monitor BFDv6 operation.

Examples

The following example enables BFDv6 debugging:

```
Router# debug ipv6 static
```

Related Commands

Command	Description
monitor event ipv6 static	Monitors the operation of the IPv6 static and IPv6 static BFDv6 neighbors using event trace.
show ipv6 static	Displays the current contents of the IPv6 routing table.

debug ipv6 wccp

To display information about IPv6 Web Cache Communication Protocol (WCCP) services, use the **debug ipv6 wccp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipv6 wccp {**default**| **vrf** *vrf-name* {**events**| **packets [control]**}| **events**| **packets** [**bypass**| **control**| **redirect**]| **platform**| **subblocks**}

no debug ipv6 wccp {**default**| **vrf** *vrf-name* {**events**| **packets [control]**}| **events**| **packets** [**bypass**| **control**| **redirect**]| **platform**| **subblocks**}

Syntax Description

default	Displays information about default WCCP services.
vrf *vrf-name*	Specifies a virtual routing and forwarding (VRF) instance to associate with a service group.
events	Displays information about significant WCCP events.
packets	Displays information about every WCCP packet received or sent by the router.
control	(Optional) Displays information about WCCP control packets.
bypass	(Optional) Displays information about WCCP bypass packets.
redirect	(Optional) Displays information about WCCP redirect packets.
platform	Displays information about the WCCP platform application programming interface (API).
subblocks	Displays information about WCCP subblocks.

Command Default

Debug information is not displayed.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.2(3)T	This command was introduced.
15.1(1)SY1	This command was integrated into Cisco IOS Release 15.1(1)SY1.

Usage Guidelines

When the **vrf** keyword is not used, the command displays debug information about all WCCP services on the router. The **default** keyword is used to specify default WCCP services.

Examples

The following is sample output from the **debug ipv6 wccp events** command when a Cisco Cache Engine is added to the list of available Web caches:

```
Router# debug ipv6 wccp events
WCCP-EVNT: Built I_See_You msg body w/1 usable web caches, change # 0000000A
WCCP-EVNT: Web Cache 2001:DB8:1::1 added
WCCP-EVNT: Built I_See_You msg body w/2 usable web caches, change # 0000000B
WCCP-EVNT: Built I_See_You msg body w/2 usable web caches, change # 0000000C
```

The following is sample output from the **debug ipv6 wccp packets** command. The router is sending keepalive packets to the Cisco Cache Engines at 2001:DB8:1::2 and 2001:DB8:1::1. Each keepalive packet has an identification number associated with it. When the Cisco Cache Engine receives a keepalive packet from the router, it sends a reply with the identification number back to the router.

```
Router# debug ipv6 wccp packets
WCCP-PKT: Received valid Here_I_Am packet from 2001:DB8:1::2 w/rcvd_id 00003532
WCCP-PKT: Sending I_See_You packet to 2001:DB8:1::2 w/ rcvd_id 00003534
WCCP-PKT: Received valid Here_I_Am packet from 2001:DB8:1::1 w/rcvd_id 00003533
WCCP-PKT: Sending I_See_You packet to 2001:DB8:1::1 w/ rcvd_id 00003535
WCCP-PKT: Received valid Here_I_Am packet from 2001:DB8:1::2 w/rcvd_id 00003534
WCCP-PKT: Sending I_See_You packet to 2001:DB8:1::2 w/ rcvd_id 00003536
WCCP-PKT: Received valid Here_I_Am packet from 2001:DB8:1::1 w/rcvd_id 00003535
WCCP-PKT: Sending I_See_You packet to 2001:DB8:1::1 w/ rcvd_id 00003537
WCCP-PKT: Received valid Here_I_Am packet from 2001:DB8:1::2 w/rcvd_id 00003536
WCCP-PKT: Sending I_See_You packet to 2001:DB8:1::2 w/ rcvd_id 00003538
WCCP-PKT: Received valid Here_I_Am packet from 2001:DB8:1::1 w/rcvd_id 00003537
WCCP-PKT: Sending I_See_You packet to 2001:DB8:1::1 w/ rcvd_id 00003539
```

Related Commands

Command	Description
clear ipv6 wccp	Clears the counter for packets redirected using WCCP.
ipv6 wccp	Enables support of the specified WCCP service for participation in a service group.
ipv6 wccp redirect	Enables packet redirection on an outbound or inbound interface using WCCP.
show ipv6 interface	Lists a summary of the IP information and status of an interface.

debug ipx ipxwan

To display debugging information for interfaces configured to use IPX wide-area network (IPXWAN), use the **debug ipx ipxwan** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipx ipxwan

no debug ipx ipxwan

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

The **debug ipx ipxwan** command is useful for verifying the startup negotiations between two routers running the IPX protocol through a WAN. This command produces output only during state changes or startup. During normal operations, no output is produced.

Examples

The following is sample output from the **debug ipx ipxwan** command during link startup:

```
Router# debug ipx ipxwan
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1, changed state to up
IPXWAN: state (Disconnect -> Sending Timer Requests) [Serial1/6666:200 (IPX line
 state brought up)]
IPXWAN: state (Sending Timer Requests -> Disconnect) [Serial1/6666:200 (IPX line
 state brought down)]
IPXWAN: state (Disconnect -> Sending Timer Requests) [Serial1/6666:200 (IPX line
 state brought up)]
IPXWAN: Send TIMER_REQ [seq 0] out Serial1/6666:200
IPXWAN: Send TIMER_REQ [seq 1] out Serial1/6666:200
IPXWAN: Send TIMER_REQ [seq 2] out Serial1/6666:200
IPXWAN: Send TIMER_REQ [seq 0] out Serial1/6666:200
IPXWAN: Rcv TIMER_REQ on Serial1/6666:200, NodeID 1234, Seq 1
IPXWAN: Send TIMER_REQ [seq 1] out Serial1/6666:200
IPXWAN: Rcv TIMER_RSP on Serial1/6666:200, NodeID 1234, Seq 1, Del 6
IPXWAN: state (Sending Timer Requests -> Master: Sent RIP/SAP) [Serial1/6666:200
 (Received Timer Response as master)]
IPXWAN: Send RIPSAP_INFO_REQ [seq 0] out Serial1/6666:200
IPXWAN: Rcv RIPSAP_INFO_RSP from Serial1/6666:200, NodeID 1234, Seq 0
IPXWAN: state (Master: Sent RIP/SAP -> Master: Connect) [Serial1/6666:200 (Received Router
 Info Rsp as Master)]
```

The following line indicates that the interface has initialized:

```
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1, changed state to up
```

The following lines indicate that the startup process failed to receive a timer response, brought the link down, then brought the link up and tried again with a new timer set:

```
IPXWAN: state (Sending Timer Requests -> Disconnect) [Serial1/6666:200 (IPX line
 state brought down)]
IPXWAN: state (Disconnect -> Sending Timer Requests) [Serial1/6666:200 (IPX line
 state brought up)]
```

The following lines indicate that the interface is sending timer requests and waiting for a timer response:

```
IPXWAN: Send TIMER_REQ [seq 0] out Serial1/6666:200
IPXWAN: Send TIMER_REQ [seq 1] out Serial1/6666:200
```

The following lines indicate that the interface has received a timer request from the other end of the link and has sent a timer response. The fourth line shows that the interface has come up as the master on the link.

```
IPXWAN: Rcv TIMER_REQ on Serial1/6666:200, NodeID 1234, Seq 1
IPXWAN: Send TIMER_REQ [seq 1] out Serial1/6666:200
IPXWAN: Rcv TIMER_RSP on Serial1/6666:200, NodeID 1234, Seq 1, Del 6
IPXWAN: state (Sending Timer Requests -> Master: Sent RIP/SAP) [Serial1/6666:200
 (Received Timer Response as master)]
```

The following lines indicate that the interface is sending RIP/SAP requests:

```
IPXWAN: Send RIPSAP_INFO_REQ [seq 0] out Serial1/6666:200
IPXWAN: Rcv RIPSAP_INFO_RSP from Serial1/6666:200, NodeID 1234, Seq 0
IPXWAN: state (Master: Sent RIP/SAP -> Master: Connect) [Serial1/6666:200 (Received Router
 Info Rsp as Master)]
```

debug ipx nasi

To display information about NetWare Asynchronous Services Interface (NASI) connections, use the **debug ipx nasi** command in Privileged EXEC configuration mode. To disable debugging output, use the **no** form of this command.

debug ipx nasi {**packets**| **error**| **activity**}

no debug ipx nasi {**packets**| **error**| **activity**}

Syntax Description

packets	Displays normal operating messages relating to incoming and outgoing NASI packets. This is the default.
error	Displays messages indicating an error or failure in the protocol processing.
activity	Displays messages relating to internal NASI processing of NASI connections. The activity option includes all NASI activity such as traffic indication, timer events, and state changes.

Command Default

Nasi protocol debugging is disabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.1	This command was introduced.

Usage Guidelines

Use the **debug ipx nasi**command to display handshake or negotiation details between Sequenced Packet Exchange (SPX), NASI protocol, and other protocols or applications. Use the **packets** option to determine the NASI traffic flow, and use the **error** option as a quick check to see why NASI connections failed.

Examples

The following is sample output from the **debug ipx nasi**command with the **packet** and **error**options.

```
Router# debug ipx nasi packet
Router# debug ipx nasi error
NASI0: 6E6E Check server info
NASI0: 6E6E sending server-info 4F00   Good response: 43 bytes
NASI0: 7A6E Query Port. Find first
NASI0: FFirst: line 0 DE, port: TTY1-__________ASYNC___^, group: ASYNC___^
```

```
NASI0: 7A6E sending Qport find-first response: 300 bytes
NASI0: 7B6E port request. setting up port
NASI: Check-login User: c h r i s
NASI: Check-login PW hash: C7 A6 C5 C7 C4 C0 C5 C3 C4 CC C5 CF C4 C8 C5 CB C4 D4 C5 D7 C4
D0 C5 D3 C4
NASI: Check-login PW: l a b
NASI1: 7B6E sending NCS Good server Data Ack in 0 bytes pkt in 13 size pkt
NASI1: 7B6E sending Preq response: 303 bytes Good
NASI1: 7B6E port request. setting up port
NASI1: 7B6E sending NCS Good server Data Ack in 0 bytes pkt in 13 size pkt
NASI1: 7B6E sending Preq response: 303 bytes Good
NASI1: 7B6E Unknown NASI code 4500 Pkt Size: 13
 45 0 0 FC 0 2 0 20 0 0 FF 1 0
NASI1: 7B6E Flush Rx Buffers
NASI1: 7B6E sending NASI server TTY data: 1 byte in 14 size pkt
NASI1: 7B6E sending NCS Good server Data Ack in 1 bytes pkt in 13 size pkt
```

In the following line:

- 0 in NASI0 is the number of the terminal (TTY) to which this NASI connection is attached.
- 0 in NASI0 is used by all NASI control connections.
- 6E6E is the associated SPX connection pointer for this NASI connection.
- Check server info is a type of incoming NASI packet.

```
NASI0: 6E6E Check server info
```

The following message indicates that the router is sending back a server-info packet with a positive acknowledgment, and the packet size is 43 bytes:

```
NASI0: 6E6E sending server-info 4F00 Good response: 43 bytes
```

The following line is a NASI packet type. Find first and Find next are NASI packet types.

```
NASI0: 7A6E Query Port. Find first
```

The following line indicates that the outgoing find first packet for the NASI connection 7A6E has line 0 DE, port name TTY1, and general name ASYNC:

```
NASI0: FFirst: line 0 DE, port: TTY1-__________ASYNC___^, group: ASYNC___^
```

The following two lines indicate:

- Received NASI packet for NASI connection in line 1. 7B6E is the NASI connection pointer. The packet code is 4500 and is not recognizable by Cisco.
- Hexadecimal dump of the packet in line 2.

```
NASI1: 7B6E Unknown NASI code 4500 Pkt Size: 13
 45 0 0 FC 0 2 0 20 0 0 FF 1 0
```

Related Commands

Command	Description
debug ipx spx	Displays debugging messages related to the SPX protocol.

debug ipx packet

To display information about packets received, sent, and forwarded, use the **debug ipx packet** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipx packet

no debug ipx packet

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

This command is useful for learning whether Internetwork Packet Exchange (IPX) packets are traveling over a router.

Note In order to generate **debug ipx packet** information on all IPX traffic traveling over the router, you must first configure the router so that fast switching is disabled. Use the **no ipx route-cache** command on all interfaces on which you want to observe traffic. If the router is configured for IPX fast switching, only non fast-switched packets will produce output. When the IPX cache is invalidated or cleared, one packet for each destination is displayed as the cache is repopulated.

Examples

The following is sample output from the **debug ipx packet** command:

```
Router# debug ipx packet
IPX: src=160.0260.8c4c.4f22, dst=1.0000.0000.0001, packet received
IPX: src=160.0260.8c4c.4f22, dst=1.0000.0000.0001,gw=183.0000.0c01.5d85,
sending packet
```

The first line indicates that the router receives a packet from a Novell station (address 160.0260.8c4c.4f22); this trace does not indicate the address of the immediate router sending the packet to this router. In the second line, the router forwards the packet toward the Novell server (address 1.0000.0000.0001) through an immediate router (183.0000.0c01.5d85).

The table below describes the significant fields shown in the display.

Table 5: debug ipx packet Field Descriptions

Field	Description
IPX	Indicates that this is an IPX packet.
src=160.0260.8c4c.4f22	Source address of the IPX packet. The Novell network number is 160. Its MAC address is 0260.8c4c.4f22.

Field	Description
dst=1.0000.0000.0001	Destination address for the IPX packet. The address 0000.0000.0001 is an internal MAC address, and the network number 1 is the internal network number of a Novell 3.11 server.
packet received	Router received this packet from a Novell station, possibly through an intermediate router.
gw=183.0000.0c01.5d85	Router is sending the packet over to the next hop router; its address of 183.0000.0c01.5d85 was learned from the IPX routing table.
sending packet	Router is attempting to send this packet.

debug ipx routing

To display information on Internetwork Packet Exchange (IPX) routing packets that the router sends and receives, use the **debug ipx routing** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipx routing {**activity**| **events**}

no debug ipx routing {**activity**| **events**}

Syntax Description

activity	Displays messages relating to IPX routing activity.
events	Displays messages relating to IPX routing events.

Command Modes

Privileged EXEC

Usage Guidelines

Normally, a router or server sends out one routing update per minute. Each routing update packet can include up to 50 entries. If many networks exist on the internetwork, the router sends out multiple packets per update. For example, if a router has 120 entries in the routing table, it would send three routing update packets per update. The first routing update packet would include the first 50 entries, the second packet would include the next 50 entries, and the last routing update packet would include the last 20 entries.

Examples

The following is sample output from the **debug ipx routing**command:

```
Router# debug ipx routing
IPXRIP: update from 9999.0260.8c6a.1733
          110801 in 1 hops, delay 2
IPXRIP: sending update to 12FF02:ffff.ffff.ffff via Ethernet 1
          network 555, metric 2, delay 3
          network 1234, metric 3, delay 4
```

The table below describes the significant fields shown in the display.

Table 6: debug ipx routing Field Descriptions

Field	Description
IPXRIP	IPX RIP packet.
update from 9999.0260.8c6a.1733	Routing update packet from an IPX server at address 9999.0260.8c6a.1733.
110801 in 1 hops	Network 110801 is one hop away from the router at address 9999.0260.8c6a.1733.

Field	Description
delay 2	Delay is a time measurement (1/18th second) that the NetWare shell uses to estimate how long to wait for a response from a file server. Also known as ticks.
sending update to 12FF02:ffff.ffff.ffff via Ethernet 1	Router is sending this IPX routing update packet to address 12FF02:ffff.ffff.ffff through Ethernet interface 1.
network 555	Packet includes routing update information for network 555.
metric 2	Network 555 is two metrics (or hops) away from the router.
delay 3	Network 555 is a delay of 3 away from the router. Delay is a measurement that the NetWare shell uses to estimate how long to wait for a response from a file server. Also known as ticks.

Related Commands

Command	Description
debug ipx sap	Displays information about IPX SAP packets.

debug ipx sap

To display information about Internetwork Packet Exchange (IPX) S ervice Advertisement Protocol (SAP) packets, use the **debug ipx sap** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipx sap [**activity**| **events**]

no debug ipx sap [**activity**| **events**]

Syntax Description

activity	(Optional) Provides more detailed output of SAP packets, including displays of services in SAP packets.
events	(Optional) Limits amount of detailed output for SAP packets to those that contain interesting events.

Command Modes

Privileged EXEC

Usage Guidelines

Normally, a router or server sends out one SAP update per minute. Each SAP packet can include up to seven entries. If many servers are advertising on the network, the router sends out multiple packets per update. For example, if a router has 20 entries in the SAP table, it would send three SAP packets per update. The first SAP would include the first seven entries, the second SAP would include the next seven entries, and the last update would include the last six entries.

Obtain the most meaningful detail by using the **debug ipx sap activity** and the **debug ipx sap events** commands together.

Caution Because the **debug ipx sap**command can generate a substantial amount of output, use it with caution on networks that have many interfaces and large service tables.

Examples

The following is sample output from the **debug ipx sap** command:

```
Router# debug ipx sap
IPXSAP: at 0023F778:
I SAP Response type 0x2 len 160 src:160.0000.0c00.070d dest:160.ffff.ffff.ffff(452)
 type 0x4, "Hello2", 199.0002.0004.0006 (451), 2 hops
 type 0x4, "Hello1", 199.0002.0004.0008 (451), 2 hops
IPXSAP: sending update to 160
IPXSAP: at 00169080:
 O SAP Update type 0x2 len 96 ssoc:0x452 dest:160.ffff.ffff.ffff(452)
IPX: type 0x4, "Magnolia", 42.0000.0000.0001 (451), 2hops
```

The **debug ipx sap** command generates multiple lines of output for each SAP packet--a packet summary message and a service detail message.

The first line displays the internal router memory address of the packet. The technical support staff may use this information in problem debugging.

```
IPXSAP: at 0023F778:
```

The table below describes the significant fields shown in the display.

Table 7: debug ipx sap Field Descriptions

Field	Description
I	Indicates whether the router received the SAP packet as input (I) or is sending an update as output (O).
SAP Response type 0x2	Packet type. Format is 0x*n*; possible values for *n* include: 1--General query 2--General response 3--Get Nearest Server request 4--Get Nearest Server response
len 160	Length of this packet (in bytes).
src: 160.000.0c00.070d	Source address of the packet.
dest:160.ffff.ffff.ffff	IPX network number and broadcast address of the destination IPX network for which the message is intended.
(452)	I PX socket number of the process sending the packet at the source address. This number is always 452, which is the socket number for the SAP process.

Field	Description
type 0x4	

Field	Description
	Indicates the type of service the server sending the packet provides. Format is 0x*n*. Some of the values for *n* are proprietary to Novell. Those values for *n*that have been published include the following (contact Novell for more information): 0--Unknown 1--User 2--User group 3--Print queue 4--File server 5--Job server 6--Gateway 7--Print server 8--Archive queue 9--Archive server A--Job queue B--Administration 21--NAS SNA gateway 24--Remote bridge server 2D--Time Synchronization VAP 2E--Dynamic SAP 47--Advertising print server 4B--Btrieve VAP 5.0 4C--SQL VAP 7A--TES--NetWare for VMS 98--NetWare access server 9A--Named Pipes server 9E--Portable NetWare--UNIX 111--Test server 166--NetWare management 233--NetWare management agent 237--NetExplorer NLM 239--HMI hub 23A--NetWare LANalyzer agent 26A--NMS management FFFF--Wildcard (any SAP service)

Field	Description
	Contact Novell for more information.
“Hello2”	Name of the server being advertised.
199.0002.0004.0006 (451)	Indicates the network number and address (and socket) of the server generating the SAP packet.
2 hops	Number of hops to the server from the router.

The fifth line of output indicates that the router sent a SAP update to network 160:

```
IPXSAP: sending update to 160
```

The format for **debug ipx sap** output describing a SAP update the router sends is similar to that describing a SAP update the router receives, except that the ssoc: field replaces the src: field, as the following line of output indicates:

O SAP Update type 0x2 len 96 ssoc:0x452 dest:160.ffff.ffff.ffff(452)

The ssoc:0x452 field indicates the IPX socket number of the process sending the packet at the source address. Possible values include the following:

451--Network Core Protocol

452--Service Advertising Protocol

453--Routing Information Protocol

455--NetBIOS

456--Diagnostics

4000 to 6000--Ephemeral sockets used for interaction with file servers and other network communications

Related Commands

Command	Description
debug ipx routing	Displays information on IPX routing packets that the router sends and receives.

debug ipx spoof

To display information about Sequenced Packet Exchange (SPX) keepalive and Internetwork Packet Exchange (IPX) watchdog packets when **ipx watchdog** and **ipx spx-spoof** are configured on the router, use the **debug ipx spoof**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipx spoof

no debug ipx spoof

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

Use this command to troubleshoot connections that use SPX spoofing when SPX keepalive spoofing is enabled.

Examples

The following is sample output from the **debug ipx spoof** command:

```
Router# debug ipx spoof

IPX: Tu1:200.0260.8c8d.da75->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7004 4B8 8 1D
23 (new) (changed:yes) Last Changed 0
IPX: Tu1:200.0260.8c8d.c558->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7104 2B8 7 29
2E (new) (changed:yes) Last Changed 0
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.c558 ln= 42 tc=02, SPX: 80 0 2B8 7104 29 7 7
 (early)
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.da75 ln= 42 tc=02, SPX: 80 0 4B8 7004 1D 8 8
 (early)
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.da75 ln= 32 tc=02, watchdog
IPX: local:200.0260.8c8d.da75->CC0001.0000.0000.0001 ln= 32 tc=00, watchdog snet
IPX: Tu1:200.0260.8c8d.da75->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7004 4B8 8 1D
23 (changed:clear) Last Changed 0
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.c558 ln= 42 tc=02, SPX: C0 0 2B8 7104 29 7 7
 (early)
IPX: Tu1:200.0260.8c8d.c558->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7104 2B8 7 29
2E (changed:clear) Last Changed 0
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.c558 ln= 42 tc=02, SPX: C0 0 2B8 7104 29 7 7
 (Last Changed 272 sec)
IPX: local:200.0260.8c8d.c558->CC0001.0000.0000.0001 ln= 42 tc=02, spx keepalive sent 80 0
 7104 2B8 7 29 2E
```

The following lines show that SPX packets were seen, but they are not seen for a connection that exists in the SPX table:

```
IPX: Tu1:200.0260.8c8d.da75->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7004 4B8 8 1D
23 (new) (changed:yes) Last Changed 0
IPX: Tu1:200.0260.8c8d.c558->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7104 2B8 7 29
2E (new) (changed:yes) Last Changed 0
```

The following lines show SPX packets for connections that exist in the SPX table but that SPX idle time has not yet elapsed and spoofing has not started:

```
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.c558 ln= 42 tc=02, SPX: 80 0 2B8 7104 29 7 7
 (early)
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.da75 ln= 42 tc=02, SPX: 80 0 4B8 7004 1D 8 8
 (early)
```

The following lines show an IPX watchdog packet and the spoofed reply:

```
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.da75 ln= 32 tc=02, watchdog
IPX: local:200.0260.8c8d.da75->CC0001.0000.0000.0001 ln= 32 tc=00, watchdog sent
```

The following lines show SPX packets that arrived more than two minutes after spoofing started. This situation occurs when the other sides of the SPX table are cleared. When the table is cleared, the routing processes stop spoofing the connection, which allows SPX keepalives from the local side to travel to the remote side and repopulate the SPX table.

```
IPX: Tu1:200.0260.8c8d.da75->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7004 4B8 8 1D
23 (changed:clear) Last Changed 0
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.c558 ln= 42 tc=02, SPX: C0 0 2B8 7104 29 7 7
 (early)
IPX: Tu1:200.0260.8c8d.c558->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7104 2B8 7 29
2E (changed:clear) Last Changed 0
```

The following lines show that an SPX keepalive packet came in and was spoofed:

```
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.c558 ln= 42 tc=02, SPX: C0 0 2B8 7104 29 7 7
 (Last Changed 272 sec)
IPX: local:200.0260.8c8d.c558->CC0001.0000.0000.0001 ln= 42 tc=02, spx keepalive sent 80 0
 7104 2B8 7 29 2E
```

debug ipx spx

To display debugging messages related to the Sequenced Packet Exchange (SPX) protocol, use the **debug ipx spx**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipx spx

no debug ipx spx

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

Use the **debug ipx spx** command to display handshaking or negotiating details between the SPX protocol and the other protocols or applications. SPX debugging messages indicate various states of SPX connections such as incoming and outgoing traffic information, timer events, and related processing of SPX connections.

Examples

The following is sample output from the **debug ipx spx**command:

```
Router# debug ipx spx
SPX: Sent an SPX packet
SPX: I Con Src/Dst 776E/20A0 d-strm 0 con-ctl 80
SPX: I Con Src/Dst 776E/20A0 d-strm FE con-ctl 40
SPX: C847C Connection close requested by peer
SPX: Sent an SPX packet
SPX: purge timer fired. Cleaning up C847C
SPX: purging spxcon C847C from conQ
SPX: returning inQ buffers
SPX: returning outQ buffers
SPX: returning unackedQ buffers
SPX: returning spxcon
SPX: I Con Src/Dst 786E/FFFF d-strm 0 con-ctl C0
SPX: new connection request for listening socket
SPX: Sent an SPX packet
SPX: I Con Src/Dst 786E/20B0 d-strm 0 con-ctl 40
SPX: 300 bytes data recvd
SPX: Sent an SPX packet
```

The following line indicates an incoming SPX packet that has a source connection ID of 776E and a destination connection ID of 20A0 (both in hexadecimal). The data stream value in the SPX packet is indicated by *d-strm*, and the connection control value in the SPX packet is indicated by *con-ctl* (both in hexadecimal). All data packets received are followed by an SPX debugging message indicating the size of the packet. All control packets received are consumed internally.

```
SPX: I Con Src/Dst 776E/20A0 d-strm 0 con-ctl 80
```

debug isdn

To display messages about activity in the structure and operation of ISDN in the Cisco IOS software, use the **debug isdn** command in privileged EXEC mode. To disable the ISDN debugging command, use the **no** form of this command.

debug isdn {**all**| **api name**| **cc** [**detail**| **interface** {**bri number**| **serial port/number**}]| **error** [**interface** {**bri number**| **serial port/number**}]| **events**| **mgmnt** [**detail**| **interface** {**bri number**| **serial port/number**}]| **q921**| **q931**| **standard** [**interface** {**bri number**| **serial port/number**}]| **tgrm**}

no debug isdn {**all**| **api name**| **cc** [**detail**| **interface** {**bri number**| **serial port/number**}]| **error** [**interface** {**bri number**| **serial port/number**}]| **events**| **mgmnt** [**detail**| **interface** {**bri number**| **serial port/number**}]| **q921**| **q931**| **standard** [**interface** {**bri number**| **serial port/number**}]| **tgrm**}

Syntax Description

all	Enables all **debug isdn** commands on all interfaces.
api *name*	Enables application programming interfaces (APIs) contained in ISDN on all interfaces. The *name* argument can be any one of the following APIs. The APIs must be entered one per command-line interface (CLI) command. To enable all of the APIs, use the **all** keyword. • **accept** --ISDN call acceptance • **all** --All ISDN API tracing • **bkhl** --ISDN backhaul API tracing • **cdapi** --ISDN API tracing • **csm** --ISDN Compact Subscriber Module API tracing • **l2sock** --ISDN Layer 2 socket API tracing • **nfas** --Non-Facility Associated Signaling • **packet** --ISDN packet API tracing • **qsig** --ISDN PRI Q Signaling API tracing • **rlm** --Redundant Link Manager API tracing
cc	Enables ISDN Call Control debug messages on all interfaces or, optionally, on a specific interface if you use the **interface** keyword. Call Control is a layer of processing within ISDN that is above the Q.931 protocol processing layer, but below the host and API layers.

detail	(Optional) Generates more information during the processing of a specific request.
interface	(Optional) Limits the **debug isdn** capability to one BRI or serial interface.
bri *number*	(Optional) Identifies a single BRI interface number (BRI 2, for example) to which the **debug isdn** command is applied.
serial *port* / *number*	(Optional) Identifies a single serial port and number (serial 1/0, for example) to which the **debug isdn** command is applied. Acceptable values are 0 through 7.
error	Generates error messages for normal exception conditions in the software on all interfaces or on a specific interface if you use the **interface** keyword. The actual significance of the message can be determined only by a detailed examination of surrounding debug messages.
events	Displays ISDN events occurring on the user side of the ISDN interface. See the **debug isdn event s** command page.
mgmnt	Enables ISDN Management Entity messages on all interfaces or, optionally, on a specific interface. Management Entity controls the activation and deactivation of Q.921 resources.
q921	Displays data link layer access procedures that are taking place at the router on the Link Access Protocol D-channel (LAPD) of its ISDN interface. See the **debug isdn q921** command page.
q931	Displays information about call setup and teardown of ISDN network connections between the local router and the network. See the **debug isdn q931** command page.
standard	Enables a selected set of **isdn debug** command messages on all interfaces or, optionally, on a specific interface if you use the **interface** keyword, that should provide sufficient information to determine why a problem is occurring.
tgrm	Displays ISDN trunk group resource manager information. See the **debug isdn tgrm**command page.

Command Default Commands are enabled on all interfaces unless a specific interface is specified.

Command Modes Privileged EXEC

Command History

Release	Modification
10.0	This command was introduced.
12.2T	This command was enhanced with the **all api cc error mgmnt**, and **standard** keywords.
12.4(6)T	The **mgmnt** keyword was enhanced to display information about sharing the terminal endpoint identifier (TEI) when the **isdn x25 dchannel q93-broadcast** command is enabled for service access point identifier (SAPI) procedures that accept X.25 calls on the BRI D channel.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines Please read the following caution before using this command.

Caution With the exception of the **debug isdn events**, **debug isdn q921**, **debug isdn q931**, and **debug isdn tgrm** commands, the commands described on this page are not intended for customer use and can cause ISDN or the Cisco IOS software to fail. The **debug isdn events**, **debug isdn q921**, **debug isdn q931**, and **debug isdn tgrm** commands are described on separate command pages.

Follow all instructions from Cisco technical support personnel when enabling and disabling these commands.

Examples The general format of the **debug isdn** command messages is as follows:

date and time: ISDN interface feature: text message

The text message can be used to determine activity in the structure and operation of ISDN in the Cisco IOS software, ISDN messages, and ISDN signaling procedures. The message must be interpreted by Cisco technical personnel.

The following example shows a typical message for the **debug isdn cc** command:

```
*Mar  1 02:29:27.751: ISDN Se1/0:23 CC: CCPRI_Go: source id 0x300, call id 0x8008, event
0x341 (pre-ccb recovery)
```

The following example enables a selected set of **debug isdn**messages that should provide sufficient information for Cisco technical personnel to determine why a problem is occurring on BRI interface 2:

```
Router# debug isdn standard interface bri 2
```

The following report (highlighted in bold for purpose of example) is displayed when the isdn x25 dchannel q931-broadcast command is used to enable sharing the TEI:

```
Router# debug isdn mgmnt
*Jun 8 22:38:56.535: ISDN BR0 Q921: User TX -> IDREQ ri=29609 ai=127
*Jun 8 22:38:56.595: ISDN BR0 Q921: User RX <- IDASSN ri=29609 ai=86
*Jun 8 22:38:56.595: ISDN BR0 SERROR: L2_Go: at bailout DLCB is NULL
L2: sapi 63 tei 127 ces 0 ev 0x3
*Jun 8 22:38:56.595: ISDN BR0 MGMNT: LM_MDL_UI_DATA_IND: message 2 ri 29609 ai 86 switch
type 9
*Jun 8 22:38:56.595: ISDN BR0 MGMNT: LM_MDL_UI_DATA_IND: OVERLAP REQUEST: ces 9 using lmtr
 tei 85 tei 85
```

debug isdn event

To display ISDN events occurring on the user side (on the router) of the ISDN interface, use the **debug isdn event** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug isdn event

no debug isdn event

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
1x.x(x)	This command was introduced.
12.4(3rd)T	This command was enhanced to display reports about SAPI 0 procedures that accept X.25 calls on the BRI D channel.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

Although the **debug isdn event** and the **debug isdn q931** commands provide similar debug information, the information is displayed in a different format. If you want to see the information in both formats, enable both commands at the same time. The displays will be intermingled.

The ISDN events that can be displayed are Q.931 events (call setup and teardown of ISDN network connections).

Use the **show dialer** command to retrieve information about the status and configuration of the ISDN interface on the router.

Use the **service timestamps debug datetime msec** global configuration command to include the time with each message.

For more information on ISDN switch types, codes, and values, see Appendix B, "ISDN Switch Types, Codes, and Values."

Examples

The following is sample output from the **debug isdn event** command of call setup events for an outgoing call:

```
Router# debug isdn event
ISDN Event: Call to 415555121202
received HOST_PROCEEDING
 Channel ID i = 0x0101
 ------------------
 Channel ID i = 0x89
received HOST_CONNECT
```

```
 Channel ID i = 0x0101
ISDN Event: Connected to 415555121202 on B1 at 64 Kb/s
```

The following shows sample **debug isdn event** output of call setup events for an incoming call. The values used for internal purposes are unpacked information elements. The values that follow the ISDN specification are an interpretation of the unpacked information elements.

```
Router# debug isdn event
received HOST_INCOMING_CALL
 Bearer Capability i = 0x080010
 -------------------
 Channel ID i = 0x0101
 Calling Party Number i = 0x0000, '415555121202'
 IE out of order or end of 'private' IEs --
 Bearer Capability i = 0x8890
 Channel ID i = 0x89
 Calling Party Number i = 0x0083, '415555121202'
ISDN Event: Received a call from 415555121202 on B1 at 64 Kb/s
ISDN Event: Accepting the call
received HOST_CONNECT
 Channel ID i = 0x0101
ISDN Event: Connected to 415555121202 on B1 at 64 Kb/s
```

The following is sample output from the **debug isdn event** command of call teardown events for a call that has been disconnected by the host side of the connection:

```
Router# debug isdn event
received HOST_DISCONNECT
ISDN Event: Call to 415555121202 was hung up
```

The following is sample output from the **debug isdn event** command of a call teardown event for an outgoing or incoming call that has been disconnected by the ISDN interface on the router side:

```
Router# debug isdn event
ISDN Event: Hangup call to call id 0x8008
```

The table below describes the significant fields shown in the display.

Table 8: debug isdn event Field Descriptions

Field	Description
Bearer Capability	Indicates the requested bearer service to be provided by the network. See Table B-4 in Appendix B, "ISDN Switch Types, Codes, and Values."
i=	Indicates the information element identifier. The value depends on the field it is associated with. Refer to the ITU-T Q.931 specification for details about the possible values associated with each field for which this identifier is relevant.

Field	Description
Channel ID	Channel Identifier. The values and corresponding channels might be identified in several ways: • Channel ID i=0x0101--Channel B1 • Channel ID i=0x0102--Channel B2 ITU-T Q.931 defines the values and channels as exclusive or preferred: • Channel ID i=0x83--Any B channel • Channel ID i=0x89--Channel B1 (exclusive) • Channel ID i=0x8A--Channel B2 (exclusive) • Channel ID i=0x81--B1 (preferred) • Channel ID i=0x82--B2 (preferred)
Calling Party Number	Identifies the called party. This field is only present in outgoing calls. The Calling Party Number field uses the IA5 character set. Note that it may be replaced by the Keypad facility field.
IE out of order or end of 'private' IEs	Indicates that an information element identifier is out of order or there are no more private network information element identifiers to interpret.
Received a call from 415555121202 on B1 at 64 Kb/s	Identifies the origin of the call. This field is present only in incoming calls. Note that the information about the incoming call includes the channel and speed. Whether the channel and speed are displayed depends on the network delivering the calling party number.

The following is sample output from the **debug isdn event** command of a call teardown event for a call that has passed call screening and then has been hung up by the ISDN interface on the far end side:

```
Router# debug isdn event
Jan  3 11:29:52.559: ISDN BR0: RX <-  DISCONNECT pd = 8  callref = 0x81
Jan  3 11:29:52.563:         Cause i = 0x8090 - Normal call clearing
```

The following is sample output from the **debug isdn event** command of a call teardown event for a call that has not passed call screening and has been rejected by the ISDN interface on the router side:

```
Router# debug isdn event
Jan  3 11:32:03.263: ISDN BR0: RX <-  DISCONNECT pd = 8  callref = 0x85
Jan  3 11:32:03.267:         Cause i = 0x8095 - Call rejected
```

The following is sample output from the **debug isdn event** command of a call teardown event for an outgoing call that uses a dialer subaddress:

```
Router# debug isdn event
```

```
Jan  3 11:41:48.483: ISDN BR0: Event: Call to 61885:1212 at 64 Kb/s
Jan  3 11:41:48.495: ISDN BR0: TX ->  SETUP pd = 8  callref = 0x04
Jan  3 11:41:48.495:         Bearer Capability i = 0x8890
Jan  3 11:41:48.499:         Channel ID i = 0x83
Jan  3 11:41:48.503:         Called Party Number i = 0x80, '61885'
Jan  3 11:41:48.507:         Called Party SubAddr i = 0x80, 'P1212'
Jan  3 11:41:48.571: ISDN BR0: RX <-  CALL_PROC pd = 8  callref = 0x84
Jan  3 11:41:48.575:         Channel ID i = 0x89
Jan  3 11:41:48.587: ISDN BR0: Event: incoming ces value = 1
Jan  3 11:41:48.587: ISDN BR0: received HOST_PROCEEDING
                        Channel ID i = 0x0101
Jan  3 11:41:48.591:    ------------------
                        Channel ID i = 0x89
Jan  3 11:41:48.731: ISDN BR0: RX <-  CONNECT pd = 8  callref = 0x84
Jan  3 11:41:48.743: ISDN BR0: Event: incoming ces value = 1
Jan  3 11:41:48.743: ISDN BR0: received HOST_CONNECT
                        Channel ID i = 0x0101
Jan  3 11:41:48.747:    ------------------
%LINK-3-UPDOWN: Interface BRI0:1 changed state to up
Jan  3 11:41:48.771: ISDN BR0: Event: Connected to 61885:1212 on B1 at 64 Kb/s
Jan  3 11:41:48.775: ISDN BR0: TX ->  CONNECT_ACK pd = 8  callref = 0x04
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0:1, changed state to up
%ISDN-6-CONNECT: Interface BRI0:1 is now connected to 61885:1212 goodie
```

The output is similar to the output of **debug isdn q931**. Refer to the **debug isdn q931**command for detailed field descriptions.

The following is sample output from the **debug isdn event** command of call setup events for a successful callback for legacy DDR:

```
Router# debug isdn event
BRI0:Caller id Callback server starting to spanky 81012345678902
: Callback timer expired
BRI0:beginning callback to spanky 81012345678902
BRI0: Attempting to dial 81012345678902
```

The following is sample output from the **debug isdn event** command for a callback that was unsuccessful because the router had no dialer map for the calling number:

```
Router# debug isdn event
BRI0:Caller id 81012345678902 callback - no matching map
```

The table below describes the significant fields shown in the display.

Table 9: debug isdn event Field Descriptions for Caller ID Callback and Legacy DDR

Field	Description
BRI0:Caller id Callback server starting to ...	Caller ID callback has started, plus host name and number called. The callback enable timer starts now.
: Callback timer expired	Callback timer has expired; callback can proceed.
BRI0:beginning callback to ... BRI0: Attempting to dial ...	Actions proceeding after the callback timer expired, plus host name and number called.

The following is sample output from the **debug isdn event** command for a callback that was successful when the dialer profiles DDR feature is configured:

```
*Mar  1 00:46:51.827: BR0:1:Caller id 81012345678901 matched to profile delorean
*Mar  1 00:46:51.827: Dialer1:Caller id Callback server starting to delorean 81012345678901
*Mar  1 00:46:54.151: : Callback timer expired
*Mar  1 00:46:54.151: Dialer1:beginning callback to delorean 81012345678901
```

```
*Mar  1 00:46:54.155: Freeing callback to delorean 81012345678901
*Mar  1 00:46:54.155: BRI0: Dialing cause Callback return call
*Mar  1 00:46:54.155: BRI0: Attempting to dial 81012345678901
*Mar  1 00:46:54.503: %LINK-3-UPDOWN: Interface BRI0:2, changed state to up
*Mar  1 00:46:54.523: %DIALER-6-BIND: Interface BRI0:2 bound to profile Dialer1
*Mar  1 00:46:55.139: %LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0:2, changed state
 to up
*Mar  1 00:46:58.187: %ISDN-6-CONNECT: Interface BRI0:2 is now connected to 81012345678901
 delorean
```

The following example provides information about accepting X.25 calls on the ISDN D channel (for purpose of example, bold type indicates messages of interest in the following output):

```
Router# debug isdn event
*Sep 28 12:34:29.747: ISDN BR1/1 EVENTd: isdn_host_packet_mode_events: Host packet call
received  call id 0xB
```

The table below describes significant fields of call setup events for a successful callback for the sample output from the **debug isdn event** command when the dialer profiles DDR feature is configured.

Table 10: debug isdn event Field Descriptions for Caller ID Callback and Dialer Profiles

Field	Description
BR0:1:Caller id ... matched to profile ...	Interface, channel number, caller ID that are matched, and the profile to bind to the interface.
: Callback timer expired	Callback timer has expired; callback can proceed.
Dialer1:beginning callback to...	Callback process is beginning to the specified number.
Freeing callback to...	Callback has been started to the specified number, and the number has been removed from the callback list.
BRI0: Dialing cause Callback return call BRI0: Attempting to dial	The reason for the call and the number being dialed.
%LINK-3-UPDOWN: Interface BRI0:2, changed state to up	Interface status: up.
%DIALER-6-BIND: Interface BRI0:2 bound to profile Dialer1	Profile bound to the interface.
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0:2, changed state to up	Line protocol status: up.
%ISDN-6-CONNECT: Interface BRI0:2 is now connected to ...	Interface is now connected to the specified host and number.
isdn_host_packet_mode_events: Host packet call received call id 0xB	Host is accepting incoming X.25 call using ITU Q.931 SAPI value 0 procedures.

Related Commands

Command	Description
debug isdn q931	Displays call setup and teardown information of ISDN Layer 3 network connections.

debug isdn q921

To display data link layer (Layer 2) access procedures that are taking place at the router on the D channel (Link Access Procedure or LAPD) of its ISDN interface, use the **debug isdn q921**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug isdn q921 [**detail**| **frame**| **interface** [**bri** *number*]]

no debug isdn q921 [**detail**| **frame**| **interface**]

Syntax Description

detail	(Optional) Displays ISDN Q.921 packet detail.
frame	(Optional) Displays ISDN Q.921 frame contents.
interface	(Optional) Specifies an interface for debugging.
bri *number*	(Optional) Specifies the BRI interface and selects the interface number. Valid values are from 0 to 6.

Command Default

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0	This command was introduced.
12.2(15)ZJ	The **detail** and **frame**keywords were added.
12.3(4)T	This command was integrated into Cisco IOS Release 12.3(4)T.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

The ISDN data link layer interface provided by the router conforms to the user interface specification defined by ITU-T recommendation Q.921. The **debug isdn q921** command output is limited to commands and responses exchanged during peer-to-peer communication carried over the D channel. This debug information does not include data transmitted over the B channels that are also part of the router ISDN interface. The peers (data link layer entities and layer management entities on the routers) communicate with each other with an ISDN switch over the D channel.

Note The ISDN switch provides the network interface defined by Q.921. This debug command does not display data link layer access procedures taking place within the ISDN network (that is, procedures taking place on the network side of the ISDN connection). Refer to Appendix B, "ISDN Switch Types, Codes, and Values ," in the *ISDN Switch Types, Codes, and Values* document on Cisco.com for a list of the supported ISDN switch types.

A router can be the calling or called party of the ISDN Q.921 data link layer access procedures. If the router is the calling party, the command displays information about an outgoing call. If the router is the called party, the command displays information about an incoming call and the keepalives.

The **debug isdn q921** command can be used with the **debug isdn event**, **debug isdn q931**, **debug isdn q921 frame**, and **debug isdn q921 detail**commands at the same time. The displays are intermingled.

Use the **service timestamps debug datetime msec** global configuration command to include the time with each message.

Examples

The following is example output for a single active data link connection (DLC). The debugs turned on are **debug isdn q921**, **debug isdn q921 frame**, and **debug isdn q921 detail**. In the debugs below, "Q921" followed by a colon (:) indicates that **debug isdn q921** has been entered. "Q921" followed by the letter "f" indicates that **debug isdn q921 frame** has been entered. "Q921" followed by the letter "d" indicates that **debug isdn q921 detail** has been entered.

The following output shows that the L2 frame is received. The first two octets form the address field; the third octet forms the control field. The address field identifies the originator of a frame and whether it is a command or a response. The second octet of the address field identifies the DLC with which the frame is associated. The control field (third octet) contains the frame type code and sequence number information.

```
00:12:10:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
00:12:10:ISDN Se1:15 Q921f:PBXb RX <- 0x0E03EF
```

The following output interprets the octet information. String "PBXb" indicates that the side receiving (RX) this frame is acting as a PBXb (as opposed to PBXa, which is the other possibility). This example also gives information about the type of frame received (SABMR), the associated DLC (1), the frame type code received from the control field (cntl=SABMR), and the sequence number (indicated by nbit, which is 0 in this case).

```
00:12:10:ISDN Se1:15 Q921:PBXb RX <- SABMR dlci=1 cntl=SABMR nbit=0
```

The following output shows information received from the driver (source_id of x200) showing an L2 frame (event x141). This results from the SABMR frame that was received from the peer PBX (v_bit and chan do not have any significance in this case).

```
00:12:10:ISDN Se1:15 Q921d:process_rxdata:Frame sent to L2
00:12:10:ISDN  Q921d:isdn_from_driver_process:event_count 3
00:12:10:ISDN Se1:15 Q921d:dpnss_l2_main:source_id x200 event x141 v_bit x0 chan x0
```

The following output shows that DPNSS L2 for DLC 1 (chan 1) has received an SABMR frame (event x0) in the IDLE state (s_dpnss_idle):

```
00:12:10:ISDN Se1:15 Q921d:s_dpnss_idle:event x0 chan 1
```

The following output shows that for DLC 1 (chan 1 above), a UA frame (event x1) needs to be sent to the driver (dest x200):

```
00:12:10:ISDN Se1:15 Q921d:dpnss_l2_mail:dest x200 event x1 v_bit 1 chan 1 out_pkt x630531A4
```

The following output shows that for DLC 1, a DL_EST_IND (event x201) needs to be sent to L3 (DUA in this case because of the backhauling) indicating that this DLC is now up (in RESET COMPLETE state):

```
00:12:10:ISDN Se1:15 Q921d:dpnss_l2_mail:dest x300 event x201 v_bit 1 chan 1 out_pkt x0
```

The following output shows that the L2 frame is transmitted (TX):

```
00:12:10:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
00:12:10:ISDN Se1:15 Q921f:PBXb TX -> 0x0E0363
```

The following output shows that string "PBXb" is the side transmitting (TX) and that this frame is acting as PBX B. This example also gives information about the associated DLC (1), the frame type code transmitted from the control field (cntl=UA), and the sequence number (indicated by nbit, which is 0 in this case).

```
00:12:10:ISDN Se1:15 Q921:PBXb TX -> UA dlci=1 cntl=UA nbit=0
```

The following is complete debugging output from a DPNSS call:

```
Jan  8 17:24:43.499:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:24:43.499:ISDN Se2/0:15 Q921f:PBXa TX -> 0x440303
Jan  8 17:24:43.499:ISDN Se2/0:15 Q921:PBXa TX -> UI(R) dlci=1 cntl=UI nbit=0
Jan  8 17:24:43.499:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
Jan  8 17:24:43.503:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:24:43.503:ISDN Se2/0:15 Q921f:PBXa RX <-
       0x44030300102A34232A35302A33333330
Jan  8 17:24:43.503:   303030312334343430303031
Jan  8 17:24:43.503:ISDN Se2/0:15 Q921:PBXa RX <- UI(C) dlci=1 cntl=UI nbit=0
       i=0x00102A34232A35302A3333333030303031234343430303031
Jan  8 17:24:43.503:ISDN Se2/0:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:24:43.503:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:24:43.507:ISDN Se2/0:15 Q921d:dpnss_l2_main:source_id x200 event
       x141 v_bit x0 chan x0
Jan  8 17:24:43.507:ISDN Se2/0:15 Q921d:s_dpnss_information_transfer:event x2
       chan 1
Jan  8 17:24:43.507:ISDN Se2/0:15 Q921d:dpnss_l2_mail:dest x200 event x3
       v_bit 1 chan 1 out_pkt x63F183D4
Jan  8 17:24:43.507:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:24:43.507:ISDN Se2/0:15 Q921f:PBXa TX -> 0x440303
Jan  8 17:24:43.507:ISDN Se2/0:15 Q921:PBXa TX -> UI(R) dlci=1 cntl=UI nbit=0
Jan  8 17:24:43.507:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
Jan  8 17:24:43.515:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:24:43.515:ISDN Se2/0:15 Q921f:PBXa RX <-
       0x44030300102A34232A35302A33333330
Jan  8 17:24:43.515:   303030312334343430303031
Jan  8 17:24:43.515:ISDN Se2/0:15 Q921:PBXa RX <- UI(C) dlci=1 cntl=UI nbit=0
       i=0x00102A34232A35302A3333333030303031234343430303031
Jan  8 17:24:43.515:ISDN Se2/0:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:24:43.515:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:24:43.515:ISDN Se2/0:15 Q921d:dpnss_l2_main:source_id x200 event
       x141 v_bit x0 chan x0
Jan  8 17:24:43.515:ISDN Se2/0:15 Q921d:s_dpnss_information_transfer:event x2
       chan 1
Jan  8 17:24:43.515:ISDN Se2/0:15 Q921d:dpnss_l2_mail:dest x200 event x3
       v_bit 1 chan 1 out_pkt x63F183D4
Jan  8 17:24:43.515:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:24:43.519:ISDN Se2/0:15 Q921f:PBXa TX -> 0x440303
Jan  8 17:24:43.519:ISDN Se2/0:15 Q921:PBXa TX -> UI(R) dlci=1 cntl=UI nbit=0
Jan  8 17:24:43.519:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
Jan  8 17:24:43.599:ISDN Se2/1:15 Q921d:dpnss_l2_main:source_id x4 event x240
       v_bit x0 chan x2
Jan  8 17:24:43.599:ISDN Se2/1:15 Q921d:s_dpnss_information_transfer:event
       x240 chan 1
Jan  8 17:24:43.599:ISDN Se2/1:15 Q921d:dpnss_l2_mail:dest x200 event x2
       v_bit 1 chan 1 out_pkt x63EE5780
Jan  8 17:24:43.599:ISDN Se2/1:15 LIFd:LIF_StartTimer:timer (0x63E569A8),
       ticks (500), event (0x1201)
Jan  8 17:24:43.599:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:24:43.599:ISDN Se2/1:15 Q921f:PBXa TX ->
       0x46030300102A31232A35302A33333330
Jan  8 17:24:43.599:   303030312334343430303031
Jan  8 17:24:43.599:ISDN Se2/1:15 Q921:PBXa TX -> UI(C) dlci=1 cntl=UI nbit=0
```

```
        i=0x00102A31232A35302A3333333030303031233434343030303031
Jan  8 17:24:43.599:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
Jan  8 17:24:43.623:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:24:43.623:ISDN Se2/1:15 Q921f:PBXa RX <- 0x460303
Jan  8 17:24:43.623:ISDN Se2/1:15 Q921:PBXa RX <- UI(R) dlci=1 cntl=UI nbit=0
Jan  8 17:24:43.623:ISDN Se2/1:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:24:43.623:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:24:43.627:ISDN Se2/1:15 Q921d:dpnss_l2_main:source_id x200 event
        x141 v_bit x0 chan x0
Jan  8 17:24:43.627:ISDN Se2/1:15 Q921d:s_dpnss_information_transfer:event x3
        chan 1
Jan  8 17:24:43.719:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:24:43.719:ISDN Se2/1:15 Q921f:PBXa RX <-
        0x440313092A34232A35302A3434343030
Jan  8 17:24:43.719:   303031232A31382A33312A33312A3331
Jan  8 17:24:43.719:   23
Jan  8 17:24:43.719:ISDN Se2/1:15 Q921:PBXa RX <- UI(C) dlci=1 cntl=UI nbit=1
        i=0x092A34232A35302A34343430303031232A31382A33312A33312A333123
Jan  8 17:24:43.719:ISDN Se2/1:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:24:43.719:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:24:43.719:ISDN Se2/1:15 Q921d:dpnss_l2_main:source_id x200 event
        x141 v_bit x0 chan x0
Jan  8 17:24:43.719:ISDN Se2/1:15 Q921d:s_dpnss_information_transfer:event x2
        chan 1
Jan  8 17:24:43.719:ISDN Se2/1:15 Q921d:dpnss_l2_mail:dest x300 event x241
        v_bit 1 chan 1 out_pkt x63EE5780
Jan  8 17:24:43.719:ISDN Se2/1:15 Q921d:dpnss_l2_mail:dest x200 event x3
        v_bit 1 chan 1 out_pkt x63EE57CC
Jan  8 17:24:43.723:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:24:43.723:ISDN Se2/1:15 Q921f:PBXa TX -> 0x440313
Jan  8 17:24:43.723:ISDN Se2/1:15 Q921:PBXa TX -> UI(R) dlci=1 cntl=UI nbit=1
Jan  8 17:24:43.723:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
Jan  8 17:24:43.727:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:24:43.727:ISDN Se2/1:15 Q921f:PBXa RX <-
        0x440313092A34232A35302A3434343030
Jan  8 17:24:43.727:   303031232A31382A33312A33312A3331
Jan  8 17:24:43.727:   23
Jan  8 17:24:43.727:ISDN Se2/1:15 Q921:PBXa RX <- UI(C) dlci=1 cntl=UI nbit=1
        i=0x092A34232A35302A34343430303031232A31382A33312A33312A333123
Jan  8 17:24:43.727:ISDN Se2/1:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:24:43.727:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:24:43.731:ISDN Se2/1:15 Q921d:dpnss_l2_main:source_id x200 event
        x141 v_bit x0 chan x0
Jan  8 17:24:43.731:ISDN Se2/1:15 Q921d:s_dpnss_information_transfer:event x2
        chan 1
Jan  8 17:24:43.731:ISDN Se2/1:15 Q921d:dpnss_l2_mail:dest x200 event x3
        v_bit 1 chan 1 out_pkt x63EE57CC
Jan  8 17:24:43.731:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:24:43.731:ISDN Se2/1:15 Q921f:PBXa TX -> 0x440313
Jan  8 17:24:43.731:ISDN Se2/1:15 Q921:PBXa TX -> UI(R) dlci=1 cntl=UI nbit=1
Jan  8 17:24:43.731:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
Jan  8 17:24:43.739:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:24:43.739:ISDN Se2/1:15 Q921f:PBXa RX <-
        0x440313092A34232A35302A3434343030
Jan  8 17:24:43.739:   303031232A31382A33312A33312A3331
Jan  8 17:24:43.739:   23
Jan  8 17:24:43.739:ISDN Se2/1:15 Q921:PBXa RX <- UI(C) dlci=1 cntl=UI nbit=1
        i=0x092A34232A35302A34343430303031232A31382A33312A33312A333123
Jan  8 17:24:43.739:ISDN Se2/1:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:24:43.739:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:24:43.739:ISDN Se2/1:15 Q921d:dpnss_l2_main:source_id x200 event
        x141 v_bit x0 chan x0
Jan  8 17:24:43.739:ISDN Se2/1:15 Q921d:s_dpnss_information_transfer:event x2
        chan 1
Jan  8 17:24:43.739:ISDN Se2/1:15 Q921d:dpnss_l2_mail:dest x200 event x3
        v_bit 1 chan 1 out_pkt x63EE57CC
Jan  8 17:24:43.739:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:24:43.743:ISDN Se2/1:15 Q921f:PBXa TX -> 0x440313
Jan  8 17:24:43.743:ISDN Se2/1:15 Q921:PBXa TX -> UI(R) dlci=1 cntl=UI nbit=1
Jan  8 17:24:43.743:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
Jan  8 17:24:43.787:ISDN Se2/0:15 Q921d:dpnss_l2_main:source_id x4 event x240
        v_bit x0 chan x2
Jan  8 17:24:43.787:ISDN Se2/0:15 Q921d:s_dpnss_information_transfer:event
```

```
        x240 chan 1
Jan  8 17:24:43.787:ISDN Se2/0:15 Q921d:dpnss_l2_mail:dest x200 event x2
        v_bit 1 chan 1 out_pkt x636B1B64
Jan  8 17:24:43.787:ISDN Se2/0:15 LIFd:LIF_StartTimer:timer (0x63A4AFBC),
        ticks (500), event (0x1201)
Jan  8 17:24:43.791:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:24:43.791:ISDN Se2/0:15 Q921f:PBXa TX ->
        0x460313092A31232A35302A3434343030
Jan  8 17:24:43.791:   30303123
Jan  8 17:24:43.791:ISDN Se2/0:15 Q921:PBXa TX -> UI(C) dlci=1 cntl=UI nbit=1
        i=0x092A31232A35302A343434303030303123
Jan  8 17:24:43.791:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
Jan  8 17:24:43.811:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:24:43.811:ISDN Se2/0:15 Q921f:PBXa RX <- 0x460313
Jan  8 17:24:43.811:ISDN Se2/0:15 Q921:PBXa RX <- UI(R) dlci=1 cntl=UI nbit=1
Jan  8 17:24:43.811:ISDN Se2/0:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:24:43.811:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:24:43.811:ISDN Se2/0:15 Q921d:dpnss_l2_main:source_id x200 event
        x141 v_bit x0 chan x0
Jan  8 17:24:43.811:ISDN Se2/0:15 Q921d:s_dpnss_information_transfer:event x3
        chan 1
Jan  8 17:24:52.107:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:24:52.107:ISDN Se2/1:15 Q921f:PBXa RX <-
        0x440303052A34232A35302A3434343030
Jan  8 17:24:52.107:   303031232A31382A33312A33312A3331
Jan  8 17:24:52.107:   23
Jan  8 17:24:52.107:ISDN Se2/1:15 Q921:PBXa RX <- UI(C) dlci=1 cntl=UI nbit=0
        i=0x052A34232A35302A343434303030031232A31382A33312A33312A333123
Jan  8 17:24:52.107:ISDN Se2/1:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:24:52.107:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:24:52.111:ISDN Se2/1:15 Q921d:dpnss_l2_main:source_id x200 event
        x141 v_bit x0 chan x0
Jan  8 17:24:52.111:ISDN Se2/1:15 Q921d:s_dpnss_information_transfer:event x2
        chan 1
Jan  8 17:24:52.111:ISDN Se2/1:15 Q921d:dpnss_l2_mail:dest x300 event x241
        v_bit 1 chan 1 out_pkt x63F19CC8
Jan  8 17:24:52.111:ISDN Se2/1:15 Q921d:dpnss_l2_mail:dest x200 event x3
        v_bit 1 chan 1 out_pkt x63F19D14
Jan  8 17:24:52.111:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:24:52.111:ISDN Se2/1:15 Q921f:PBXa TX -> 0x440303
Jan  8 17:24:52.111:ISDN Se2/1:15 Q921:PBXa TX -> UI(R) dlci=1 cntl=UI nbit=0
Jan  8 17:24:52.111:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
Jan  8 17:24:52.119:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:24:52.119:ISDN Se2/1:15 Q921f:PBXa RX <-
        0x440303052A34232A35302A3434343030
Jan  8 17:24:52.119:   303031232A31382A33312A33312A3331
Jan  8 17:24:52.119:   23
Jan  8 17:24:52.119:ISDN Se2/1:15 Q921:PBXa RX <- UI(C) dlci=1 cntl=UI nbit=0
        i=0x052A34232A35302A343434303030031232A31382A33312A33312A333123
Jan  8 17:24:52.119:ISDN Se2/1:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:24:52.119:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:24:52.119:ISDN Se2/1:15 Q921d:dpnss_l2_main:source_id x200 event
        x141 v_bit x0 chan x0
        x141 v_bit x0 chan x0
Jan  8 17:24:52.119:ISDN Se2/1:15 Q921d:s_dpnss_information_transfer:event x2
        chan 1
Jan  8 17:24:52.119:ISDN Se2/1:15 Q921d:dpnss_l2_mail:dest x200 event x3
        v_bit 1 chan 1 out_pkt x63F19D14
Jan  8 17:24:52.119:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:24:52.123:ISDN Se2/1:15 Q921f:PBXa TX -> 0x440303
Jan  8 17:24:52.123:ISDN Se2/1:15 Q921:PBXa TX -> UI(R) dlci=1 cntl=UI nbit=0
Jan  8 17:24:52.123:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
Jan  8 17:24:52.127:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:24:52.127:ISDN Se2/1:15 Q921f:PBXa RX <-
        0x440303052A34232A35302A3434343030
Jan  8 17:24:52.127:   303031232A31382A33312A33312A3331
Jan  8 17:24:52.127:   23
Jan  8 17:24:52.127:ISDN Se2/1:15 Q921:PBXa RX <- UI(C) dlci=1 cntl=UI nbit=0
        i=0x052A34232A35302A343434303030031232A31382A33312A33312A333123
Jan  8 17:24:52.127:ISDN Se2/1:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:24:52.127:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:24:52.131:ISDN Se2/1:15 Q921d:dpnss_l2_main:source_id x200 event
        x141 v_bit x0 chan x0
```

```
Jan  8 17:24:52.131:ISDN Se2/1:15 Q921d:s_dpnss_information_transfer:event x2
       chan 1
Jan  8 17:24:52.131:ISDN Se2/1:15 Q921d:dpnss_l2_mail:dest x200 event x3
       v_bit 1 chan 1 out_pkt x63F19D14
Jan  8 17:24:52.131:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:24:52.131:ISDN Se2/1:15 Q921f:PBXa TX -> 0x440303
Jan  8 17:24:52.131:ISDN Se2/1:15 Q921:PBXa TX -> UI(R) dlci=1 cntl=UI nbit=0
Jan  8 17:24:52.131:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
Jan  8 17:24:52.159:ISDN Se2/0:15 Q921d:dpnss_l2_main:source_id x4 event x240
       v_bit x0 chan x2
Jan  8 17:24:52.159:ISDN Se2/0:15 Q921d:s_dpnss_information_transfer:event
       x240 chan 1
Jan  8 17:24:52.159:ISDN Se2/0:15 Q921d:dpnss_l2_mail:dest x200 event x2
       v_bit 1 chan 1 out_pkt x63F19CC8
Jan  8 17:24:52.159:ISDN Se2/0:15 LIFd:LIF_StartTimer:timer (0x63A4AFBC),
       ticks (500), event (0x1201)
Jan  8 17:24:52.159:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:24:52.159:ISDN Se2/0:15 Q921f:PBXa TX ->
       0x460303052A35302A3434343030303031
Jan  8 17:24:52.159:   23
Jan  8 17:24:52.159:ISDN Se2/0:15 Q921:PBXa TX -> UI(C) dlci=1 cntl=UI nbit=0
       i=0x052A35302A343434303030303123
Jan  8 17:24:52.159:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
Jan  8 17:24:52.179:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:24:52.179:ISDN Se2/0:15 Q921f:PBXa RX <- 0x460303
Jan  8 17:24:52.179:ISDN Se2/0:15 Q921:PBXa RX <- UI(R) dlci=1 cntl=UI nbit=0
Jan  8 17:24:52.179:ISDN Se2/0:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:24:52.183:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:24:52.183:ISDN Se2/0:15 Q921d:dpnss_l2_main:source_id x200 event
       x141 v_bit x0 chan x0
Jan  8 17:24:52.183:ISDN Se2/0:15 Q921d:s_dpnss_information_transfer:event x3
       chan 1
Jan  8 17:25:31.811:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:25:31.811:ISDN Se2/0:15 Q921f:PBXa RX <- 0x4403130830
Jan  8 17:25:31.811:ISDN Se2/0:15 Q921:PBXa RX <- UI(C) dlci=1 cntl=UI nbit=1
       i=0x0830
Jan  8 17:25:31.811:ISDN Se2/0:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:25:31.811:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:25:31.811:ISDN Se2/0:15 Q921d:dpnss_l2_main:source_id x200 event
       x141 v_bit x0 chan x0
Jan  8 17:25:31.811:ISDN Se2/0:15 Q921d:s_dpnss_information_transfer:event x2
       chan 1
Jan  8 17:25:31.811:ISDN Se2/0:15 Q921d:dpnss_l2_mail:dest x300 event x241
       v_bit 1 chan 1 out_pkt x63F1806C
Jan  8 17:25:31.811:ISDN Se2/0:15 Q921d:dpnss_l2_mail:dest x200 event x3
       v_bit 1 chan 1 out_pkt x636710B8
Jan  8 17:25:31.815:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:25:31.815:ISDN Se2/0:15 Q921f:PBXa TX -> 0x440313
Jan  8 17:25:31.815:ISDN Se2/0:15 Q921:PBXa TX -> UI(R) dlci=1 cntl=UI nbit=1
Jan  8 17:25:31.815:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
Jan  8 17:25:31.819:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:25:31.819:ISDN Se2/0:15 Q921f:PBXa RX <- 0x4403130830
Jan  8 17:25:31.819:ISDN Se2/0:15 Q921:PBXa RX <- UI(C) dlci=1 cntl=UI nbit=1
       i=0x0830
Jan  8 17:25:31.819:ISDN Se2/0:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:25:31.819:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:25:31.823:ISDN Se2/0:15 Q921d:dpnss_l2_main:source_id x200 event
       x141 v_bit x0 chan x0
Jan  8 17:25:31.823:ISDN Se2/0:15 Q921d:s_dpnss_information_transfer:event x2
       chan 1
Jan  8 17:25:31.823:ISDN Se2/0:15 Q921d:dpnss_l2_mail:dest x200 event x3
       v_bit 1 chan 1 out_pkt x63F19CC8
Jan  8 17:25:31.823:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:25:31.823:ISDN Se2/0:15 Q921f:PBXa TX -> 0x440313
Jan  8 17:25:31.823:ISDN Se2/0:15 Q921:PBXa TX -> UI(R) dlci=1 cntl=UI nbit=1
Jan  8 17:25:31.823:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
Jan  8 17:25:31.831:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:25:31.831:ISDN Se2/0:15 Q921f:PBXa RX <- 0x4403130830
Jan  8 17:25:31.831:ISDN Se2/0:15 Q921:PBXa RX <- UI(C) dlci=1 cntl=UI nbit=1
       i=0x0830
Jan  8 17:25:31.831:ISDN Se2/0:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:25:31.831:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:25:31.831:ISDN Se2/0:15 Q921d:dpnss_l2_main:source_id x200 event
```

```
        x141 v_bit x0 chan x0
Jan  8 17:25:31.831:ISDN Se2/0:15 Q921d:s_dpnss_information_transfer:event x2
        chan 1
Jan  8 17:25:31.831:ISDN Se2/0:15 Q921d:dpnss_l2_mail:dest x200 event x3
        v_bit 1 chan 1 out_pkt x636710B8
Jan  8 17:25:31.835:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:25:31.835:ISDN Se2/0:15 Q921f:PBXa TX -> 0x440313
Jan  8 17:25:31.835:ISDN Se2/0:15 Q921:PBXa TX -> UI(R) dlci=1 cntl=UI nbit=1
Jan  8 17:25:31.835:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
Jan  8 17:25:31.851:ISDN Se2/1:15 Q921d:dpnss_l2_main:source_id x4 event x240
        v_bit x0 chan x2
Jan  8 17:25:31.851:ISDN Se2/1:15 Q921d:s_dpnss_information_transfer:event
        x240 chan 1
Jan  8 17:25:31.851:ISDN Se2/1:15 Q921d:dpnss_l2_mail:dest x200 event x2
        v_bit 1 chan 1 out_pkt x63F1806C
Jan  8 17:25:31.851:ISDN Se2/1:15 LIFd:LIF_StartTimer:timer (0x63E569A8),
        ticks (500), event (0x1201)
Jan  8 17:25:31.851:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:25:31.855:ISDN Se2/1:15 Q921f:PBXa TX -> 0x4603130830
Jan  8 17:25:31.855:ISDN Se2/1:15 Q921:PBXa TX -> UI(C) dlci=1 cntl=UI nbit=1
        i=0x0830
Jan  8 17:25:31.855:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
Jan  8 17:25:31.875:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:25:31.875:ISDN Se2/1:15 Q921f:PBXa RX <- 0x460313
Jan  8 17:25:31.875:ISDN Se2/1:15 Q921:PBXa RX <- UI(R) dlci=1 cntl=UI nbit=1
Jan  8 17:25:31.875:ISDN Se2/1:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:25:31.875:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:25:31.875:ISDN Se2/1:15 Q921d:dpnss_l2_main:source_id x200 event
        x141 v_bit x0 chan x0
Jan  8 17:25:31.875:ISDN Se2/1:15 Q921d:s_dpnss_information_transfer:event x3
        chan 1
Jan  8 17:25:31.879:ISDN Se2/0:15 Q921d:dpnss_l2_main:source_id x4 event x240
        v_bit x0 chan x2
Jan  8 17:25:31.879:ISDN Se2/0:15 Q921d:s_dpnss_information_transfer:event
        x240 chan 1
Jan  8 17:25:31.879:ISDN Se2/0:15 Q921d:dpnss_l2_mail:dest x200 event x2
        v_bit 1 chan 1 out_pkt x63EFC5AC
Jan  8 17:25:31.879:ISDN Se2/0:15 LIFd:LIF_StartTimer:timer (0x63A4AFBC),
        ticks (500), event (0x1201)
Jan  8 17:25:31.879:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:25:31.879:ISDN Se2/0:15 Q921f:PBXa TX -> 0x4603130830
Jan  8 17:25:31.879:ISDN Se2/0:15 Q921:PBXa TX -> UI(C) dlci=1 cntl=UI nbit=1
        i=0x0830
Jan  8 17:25:31.883:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
Jan  8 17:25:31.899:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:25:31.899:ISDN Se2/0:15 Q921f:PBXa RX <- 0x460313
Jan  8 17:25:31.899:ISDN Se2/0:15 Q921:PBXa RX <- UI(R) dlci=1 cntl=UI nbit=1
Jan  8 17:25:31.899:ISDN Se2/0:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:25:31.899:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:25:31.903:ISDN Se2/0:15 Q921d:dpnss_l2_main:source_id x200 event
        x141 v_bit x0 chan x0
Jan  8 17:25:31.903:ISDN Se2/0:15 Q921d:s_dpnss_information_transfer:event x3
        chan 1
Jan  8 17:25:32.063:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:25:32.063:ISDN Se2/1:15 Q921f:PBXa RX <- 0x4403130830
Jan  8 17:25:32.063:ISDN Se2/1:15 Q921:PBXa RX <- UI(C) dlci=1 cntl=UI nbit=1
        i=0x0830
Jan  8 17:25:32.063:ISDN Se2/1:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:25:32.063:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:25:32.067:ISDN Se2/1:15 Q921d:dpnss_l2_main:source_id x200 event
        x141 v_bit x0 chan x0
Jan  8 17:25:32.067:ISDN Se2/1:15 Q921d:s_dpnss_information_transfer:event x2
        chan 1
Jan  8 17:25:32.067:ISDN Se2/1:15 Q921d:dpnss_l2_mail:dest x300 event x241
        v_bit 1 chan 1 out_pkt x63EFC5AC
Jan  8 17:25:32.067:ISDN Se2/1:15 Q921d:dpnss_l2_mail:dest x200 event x3
        v_bit 1 chan 1 out_pkt x6367175C
Jan  8 17:25:32.067:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:25:32.067:ISDN Se2/1:15 Q921f:PBXa TX -> 0x440313
Jan  8 17:25:32.067:ISDN Se2/1:15 Q921:PBXa TX -> UI(R) dlci=1 cntl=UI nbit=1
Jan  8 17:25:32.067:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
Jan  8 17:25:32.075:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:25:32.075:ISDN Se2/1:15 Q921f:PBXa RX <- 0x4403130830
```

```
Jan  8 17:25:32.075:ISDN Se2/1:15 Q921:PBXa RX <- UI(C) dlci=1 cntl=UI nbit=1
        i=0x0830
Jan  8 17:25:32.075:ISDN Se2/1:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:25:32.075:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:25:32.075:ISDN Se2/1:15 Q921d:dpnss_l2_main:source_id x200 event
        x141 v_bit x0 chan x0
Jan  8 17:25:32.075:ISDN Se2/1:15 Q921d:s_dpnss_information_transfer:event x2
        chan 1
Jan  8 17:25:32.075:ISDN Se2/1:15 Q921d:dpnss_l2_mail:dest x200 event x3
        v_bit 1 chan 1 out_pkt x6367175C
Jan  8 17:25:32.075:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:25:32.075:ISDN Se2/1:15 Q921f:PBXa TX -> 0x440313
Jan  8 17:25:32.079:ISDN Se2/1:15 Q921:PBXa TX -> UI(R) dlci=1 cntl=UI nbit=1
Jan  8 17:25:32.079:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
Jan  8 17:25:32.083:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:25:32.083:ISDN Se2/1:15 Q921f:PBXa RX <- 0x4403130830
Jan  8 17:25:32.083:ISDN Se2/1:15 Q921:PBXa RX <- UI(C) dlci=1 cntl=UI nbit=1
        i=0x0830
Jan  8 17:25:32.083:ISDN Se2/1:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:25:32.083:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:25:32.087:ISDN Se2/1:15 Q921d:dpnss_l2_main:source_id x200 event
        x141 v_bit x0 chan x0
Jan  8 17:25:32.087:ISDN Se2/1:15 Q921d:s_dpnss_information_transfer:event x2
        chan 1
Jan  8 17:25:32.087:ISDN Se2/1:15 Q921d:dpnss_l2_mail:dest x200 event x3
        v_bit 1 chan 1 out_pkt x6367175C
Jan  8 17:25:32.087:ISDN  Q921d:isdn_l2d_srq_process:QUEUE_EVENT
Jan  8 17:25:32.087:ISDN Se2/1:15 Q921f:PBXa TX -> 0x440313
Jan  8 17:25:32.087:ISDN Se2/1:15 Q921:PBXa TX -> UI(R) dlci=1 cntl=UI nbit=1
Jan  8 17:25:32.087:ISDN  Q921d:isdn_l2d_srq_process:event_count 1
```

The following output shows details of the preceding debugging events.

The first two octets (0x4403) form the address field, while the third octet (0x03) is the control field. All the octets starting from the fourth constitute DPNSS L3 information, which needs to be backhauled to the Cisco PGW2200.

```
Jan  8 17:24:43.495:ISDN  Q921d:isdn_from_driver_process:QUEUE_EVENT
Jan  8 17:24:43.495:ISDN Se2/0:15 Q921f:PBXa RX <- 0x44030300102A34232A35302A33333330
Jan  8 17:24:43.495:   303030312334343430303031
```

All of the octets following “i=” constitute DPNSS L3 information received from the peer:

```
Jan  8 17:24:43.495:ISDN Se2/0:15 Q921:PBXa RX <- UI(C) dlci=1 cntl=UI nbit=0
        i=0x00102A34232A35302A333333303030303123343434303030303031
```

In the INFORMATION TRANSFER state, DLC 1 received a UI(C) frame (event x2) from the peer carrying DPNSS L3 information:

```
Jan  8 17:24:43.495:ISDN Se2/0:15 Q921d:process_rxdata:Frame sent to L2
Jan  8 17:24:43.495:ISDN  Q921d:isdn_from_driver_process:event_count 1
Jan  8 17:24:43.495:ISDN Se2/0:15 Q921d:dpnss_l2_main:source_id x200 event
        x141 v_bit x0 chan x0
Jan  8 17:24:43.495:ISDN Se2/0:15 Q921d:s_dpnss_information_transfer:event x2 chan 1
```

For DLC 1, event information is sent to L3 (IUA BACKHAUL, indicated by dest x300). In this case, DL_DATA_IND (event x241) indicates that some L3 information has been received from the peer.

```
Jan  8 17:24:43.495:ISDN Se2/0:15 Q921d:dpnss_l2_mail:dest x300 event x241
        v_bit 1 chan 1 out_pkt x6367175C
```

Information is sent to the driver (dest x200), which is then sent to the peer): An Unnumbered Information--Response [UI(R)] (event x3) acknowledges the received Unnumbered Information--Command [UI(C)].

```
Jan  8 17:24:43.495:ISDN Se2/0:15 Q921d:dpnss_l2_mail:dest x200 event x3
        v_bit 1 chan 1 out_pkt x63F183D4
```

The following is sample output from the **debug isdn q921** command for an outgoing call:

```
Router# debug isdn q921
Jan  3 14:52:24.475: ISDN BR0: TX ->  INFOc sapi = 0  tei = 64  ns = 5  nr = 2
                              i = 0x08010705040288901801837006803631383835
Jan  3 14:52:24.503: ISDN BR0: RX <-  RRr sapi = 0  tei = 64  nr = 6
Jan  3 14:52:24.527: ISDN BR0: RX <-  INFOc sapi = 0  tei = 64  ns = 2  nr = 6
                              i = 0x08018702180189
Jan  3 14:52:24.535: ISDN BR0: TX ->  RRr sapi = 0  tei = 64  nr = 3
Jan  3 14:52:24.643: ISDN BR0: RX <-  INFOc sapi = 0  tei = 64  ns = 3  nr = 6
                              i = 0x08018707
Jan  3 14:52:24.655: ISDN BR0: TX ->  RRr sapi = 0  tei = 64  nr = 4
%LINK-3-UPDOWN: Interface BRI0:1, changed state to up
Jan  3 14:52:24.683: ISDN BR0: TX ->  INFOc sapi = 0  tei = 64  ns = 6  nr = 4
                              i = 0x0801070F
Jan  3 14:52:24.699: ISDN BR0: RX <-  RRr sapi = 0  tei = 64  nr = 7
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0:1, changed state to up
%ISDN-6-CONNECT: Interface BRI0:1 is now connected to 61885 goodie
Jan  3 14:52:34.415: ISDN BR0: RX <-  RRp sapi = 0  tei = 64 nr = 7
Jan  3 14:52:34.419: ISDN BR0: TX ->  RRf sapi = 0  tei = 64  nr = 4
```

In the following lines, the seventh and eighth most significant hexadecimal numbers indicate the type of message. 0x05 indicates a Call Setup message, 0x02 indicates a Call Proceeding message, 0x07 indicates a Call Connect message, and 0x0F indicates a Connect Ack message.

```
Jan  3 14:52:24.475: ISDN BR0: TX ->  INFOc sapi = 0  tei = 64  ns = 5  nr = 2
                              i = 0x08010705040288901801837006803631383835
Jan  3 14:52:24.527: ISDN BR0: RX <-  INFOc sapi = 0  tei = 64  ns = 2  nr = 6
                              i = 0x08018702180189
Jan  3 14:52:24.643: ISDN BR0: RX <-  INFOc sapi = 0  tei = 64  ns = 3  nr = 6
                              i = 0x08018707
Jan  3 14:52:24.683: ISDN BR0: TX ->  INFOc sapi = 0  tei = 64  ns = 6  nr = 4
                              i = 0x0801070F
```

The following is sample output from the **debug isdn q921** command for a startup message on a DMS-100 switch:

```
Router# debug isdn q921
Jan  3 14:47:28.455: ISDN BR0: RX <-  IDCKRQ  ri = 0  ai = 127 0
Jan  3 14:47:30.171: ISDN BR0: TX ->  IDREQ  ri = 31815  ai = 127
Jan  3 14:47:30.219: ISDN BR0: RX <-  IDASSN  ri = 31815  ai = 64
Jan  3 14:47:30.223: ISDN BR0: TX ->  SABMEp sapi = 0  tei = 64
Jan  3 14:47:30.227: ISDN BR0: RX <-  IDCKRQ  ri = 0  ai = 127
Jan  3 14:47:30.235: ISDN BR0: TX ->  IDCKRP  ri = 16568  ai = 64
Jan  3 14:47:30.239: ISDN BR0: RX <-  UAf sapi = 0  tei = 64
Jan  3 14:47:30.247: ISDN BR0: TX ->  INFOc sapi = 0  tei = 64  ns = 0  nr = 0
                              i = 0x08007B3A03313233
Jan  3 14:47:30.267: ISDN BR0: RX <-  RRr sapi = 0  tei = 64  nr = 1
Jan  3 14:47:34.243: ISDN BR0: TX ->  INFOc sapi = 0  tei = 64  ns = 1  nr = 0
                              i = 0x08007B3A03313233
Jan  3 14:47:34.267: ISDN BR0: RX <-  RRr sapi = 0  tei = 64  nr = 2
Jan  3 14:47:43.815: ISDN BR0: RX <-  RRp sapi = 0  tei = 64 nr = 2
Jan  3 14:47:43.819: ISDN BR0: TX ->  RRf sapi = 0  tei = 64  nr = 0
Jan  3 14:47:53.819: ISDN BR0: TX ->  RRp sapi = 0  tei = 64 nr = 0
```

The first seven lines of this example indicate a Layer 2 link establishment.

The following lines indicate the message exchanges between the data link layer entity on the local router (user side) and the assignment source point (ASP) on the network side during the TEI assignment procedure. This assumes that the link is down and no TEI currently exists.

```
Jan  3 14:47:30.171: ISDN BR0: TX ->  IDREQ  ri = 31815  ai = 127
Jan  3 14:47:30.219: ISDN BR0: RX <-  IDASSN  ri = 31815  ai = 64
```

At 14:47:30.171, the local router data link layer entity sent an Identity Request message to the network data link layer entity to request a TEI value that can be used in subsequent communication between the peer data link layer entities. The request includes a randomly generated reference number (31815) to differentiate among user devices that request automatic TEI assignment and an action indicator of 127 to indicate that the ASP can assign any TEI value available. The ISDN user interface on the router uses automatic TEI assignment.

At 14:47:30.219, the network data link entity responds to the Identity Request message with an Identity Assigned message. The response includes the reference number (31815) previously sent in the request and TEI value (64) assigned by the ASP.

The following lines indicate the message exchanges between the layer management entity on the network and the layer management entity on the local router (user side) during the TEI check procedure:

```
Jan  3 14:47:30.227: ISDN BR0: RX <-  IDCKRQ  ri = 0  ai = 127
Jan  3 14:47:30.235: ISDN BR0: TX ->  IDCKRP  ri = 16568  ai = 64
```

At 14:47:30.227, the layer management entity on the network sends the Identity Check Request message to the layer management entity on the local router to check whether a TEI is in use. The message includes a reference number that is always 0 and the TEI value to check. In this case, an ai value of 127 indicates that all TEI values should be checked. At 14:47:30.227, the layer management entity on the local router responds with an Identity Check Response message indicating that TEI value 64 is currently in use.

The following lines indicate the messages exchanged between the data link layer entity on the local router (user side) and the data link layer on the network side to place the network side into modulo 128 multiple frame acknowledged operation. Note that the data link layer entity on the network side also can initiate the exchange.

```
Jan  3 14:47:30.223: ISDN BR0: TX ->  SABMEp sapi = 0  tei = 64
Jan  3 14:47:30.239: ISDN BR0: RX <-  UAf sapi = 0  tei = 64
```

At 14:47:30.223, the data link layer entity on the local router sends the SABME command with a SAPI of 0 (call control procedure) for TEI 64. At 14:47:30.239, the first opportunity, the data link layer entity on the network responds with a UA response. This response indicates acceptance of the command. The data link layer entity sending the SABME command may need to send it more than once before receiving a UA response.

The following lines indicate the status of the data link layer entities. Both are ready to receive I frames.

```
Jan  3 14:47:43.815: ISDN BR0: RX <-  RRp sapi = 0  tei = 64 nr = 2
Jan  3 14:47:43.819: ISDN BR0: TX ->  RRf sapi = 0  tei = 64  nr = 0
```

These I-frames are typically exchanged every 10 seconds (T203 timer).

The following is sample output from the **debug isdn q921** command for an incoming call. It is an incoming SETUP message that assumes that the Layer 2 link is already established to the other side.

```
Router# debug isdn q921
Jan  3 14:49:22.507: ISDN BR0: TX ->  RRp sapi = 0  tei = 64 nr = 0
Jan  3 14:49:22.523: ISDN BR0: RX <-  RRf sapi = 0  tei = 64  nr = 2
Jan  3 14:49:32.527: ISDN BR0: TX ->  RRp sapi = 0  tei = 64 nr = 0
Jan  3 14:49:32.543: ISDN BR0: RX <-  RRf sapi = 0  tei = 64  nr = 2
Jan  3 14:49:42.067: ISDN BR0: RX <-  RRp sapi = 0  tei = 64 nr = 2
Jan  3 14:49:42.071: ISDN BR0: TX ->  RRf sapi = 0  tei = 64  nr = 0
Jan  3 14:49:47.307: ISDN BR0: RX <-  UI sapi = 0  tei = 127
                              i = 0x08011F0504028890180189700 6C13631383836
%LINK-3-UPDOWN: Interface BRI0:1, changed state to up
Jan  3 14:49:47.347: ISDN BR0: TX ->  INFOc sapi = 0  tei = 64  ns = 2  nr = 0
                              i = 0x08019F07180189
Jan  3 14:49:47.367: ISDN BR0: RX <-  RRr sapi = 0  tei = 64  nr = 3
Jan  3 14:49:47.383: ISDN BR0: RX <-  INFOc sapi = 0  tei = 64  ns = 0  nr = 3
                              i = 0x08011F0F180189
Jan  3 14:49:47.391: ISDN BR0: TX ->  RRr sapi = 0  tei = 64  nr = 1
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0:1, changed state to up
```

The table below describes the significant fields shown in the display.

Table 11: debug isdn q921 Field Descriptions

Field	Description
Jan 3 14:49:47.391	Indicates the date and time at which the frame was sent from or received by the data link layer entity on the router. The time is maintained by an internal clock.
TX	Indicates that this frame is being sent from the ISDN interface on the local router (user side).
RX	Indicates that this frame is being received by the ISDN interface on the local router from the peer (network side).
IDREQ	Indicates the Identity Request message type sent from the local router to the network (ASP) during the automatic TEI assignment procedure. This message is sent in a UI command frame. The SAPI value for this message type is always 63 (indicating that it is a Layer 2 management procedure) but it is not displayed. The TEI value for this message type is 127 (indicating that it is a broadcast operation).
ri = 31815	Indicates the Reference number used to differentiate between user devices requesting TEI assignment. This value is a randomly generated number from 0 to 65535. The same ri value sent in the IDREQ message should be returned in the corresponding IDASSN message. Note that a Reference number of 0 indicates that the message is sent from the network side management layer entity and a reference number has not been generated.
ai = 127	Indicates the Action indicator used to request that the ASP assign any TEI value. It is always 127 for the broadcast TEI. Note that in some message types, such as IDREM, a specific TEI value is indicated.
IDREM	Indicates the Identity Remove message type sent from the ASP to the user side layer management entity during the TEI removal procedure. This message is sent in a UI command frame. The message includes a reference number that is always 0, because it is not responding to a request from the local router. The ASP sends the Identity Remove message twice to avoid message loss.

Field	Description
IDASSN	Indicates the Identity Assigned message type sent from the ISDN service provider on the network to the local router during the automatic TEI assignment procedure. This message is sent in a UI command frame. The SAPI value for this message type is always 63 (indicating that it is a Layer 2 management procedure). The TEI value for this message type is 127 (indicating it is a broadcast operation).
ai = 64	Indicates the TEI value automatically assigned by the ASP. This TEI value is used by data link layer entities on the local router in subsequent communication with the network. The valid values are in the range from 64 to 126.
SABME	Indicates the set asynchronous balanced mode extended command. This command places the recipient into modulo 128 multiple frame acknowledged operation. This command also indicates that all exception conditions have been cleared. The SABME command is sent once a second for N200 times (typically three times) until its acceptance is confirmed with a UA response. For a list and brief description of other commands and responses that can be exchanged between the data link layer entities on the local router and the network, see ITU-T Recommendation Q.921.
sapi = 0	Identifies the service access point at which the data link layer entity provides services to Layer 3 or to the management layer. A SAPI with the value 0 indicates it is a call control procedure. Note that the Layer 2 management procedures such as TEI assignment, TEI removal, and TEI checking, which are tracked with the **debug isdn q921**command, do not display the corresponding SAPI value; it is implicit. If the SAPI value were displayed, it would be 63.
tei = 64	Indicates the TEI value automatically assigned by the ASP. This TEI value will be used by data link layer entities on the local router in subsequent communication with the network. The valid values are in the range from 64 to 126.

Field	Description
IDCKRQ	Indicates the Identity Check Request message type sent from the ISDN service provider on the network to the local router during the TEI check procedure. This message is sent in a UI command frame. The ri field is always 0. The ai field for this message contains either a specific TEI value for the local router to check or 127, which indicates that the local router should check all TEI values. For a list and brief description of other message types that can be exchanged between the local router and the ISDN service provider on the network, see Appendix B, "ISDN Switch Types, Codes, and Values."
IDCKRP	Indicates the Identity Check Response message type sent from the local router to the ISDN service provider on the network during the TEI check procedure. This message is sent in a UI command frame in response to the IDCKRQ message. The ri field is a randomly generated number from 0 to 65535. The ai field for this message contains the specific TEI value that has been checked.
UAf	Confirms that the network side has accepted the SABME command previously sent by the local router. The final bit is set to 1.
INFOc	Indicates that this is an Information command. It is used to transfer sequentially numbered frames containing information fields that are provided by Layer 3. The information is transferred across a data-link connection.
INFORMATION pd = 8 callref = (null)	Indicates the information fields provided by Layer 3. The information is sent one frame at a time. If multiple frames need to be sent, several Information commands are sent. The pd value is the protocol discriminator. The value 8 indicates it is call control information. The call reference number is always null for SPID information.
SPID information i = 0x343135393033383336363031	Indicates the SPID. The local router sends this information to the ISDN switch to indicate the services to which it subscribes. SPIDs are assigned by the service provider and are usually 10-digit telephone numbers followed by optional numbers. Currently, only the DMS-100 switch supports SPIDs, one for each B channel. If SPID information is sent to a switch type other than DMS-100, an error may be displayed in the debug information.

Field	Description
ns = 0	Indicates the send sequence number of sent I frames.
nr = 0	Indicates the expected send sequence number of the next received I frame. At time of transmission, this value should be equal to the value of ns. The value of nr is used to determine whether frames need to be re-sent for recovery.
RRr	Indicates the Receive Ready response for unacknowledged information transfer. The RRr is a response to an INFOc.
RRp	Indicates the Receive Ready command with the poll bit set. The data link layer entity on the user side uses the poll bit in the frame to solicit a response from the peer on the network side.
RRf	Indicates the Receive Ready response with the final bit set. The data link layer entity on the network side uses the final bit in the frame to indicate a response to the poll.
sapi	Indicates the service access point identifier. The SAPI is the point at which data link services are provided to a network layer or management entity. Currently, this field can have the value 0 (for call control procedure) or 63 (for Layer 2 management procedures).
tei	Indicates the terminal endpoint identifier (TEI) that has been assigned automatically by the assignment source point (ASP) (also called the layer management entity on the network side). The valid range is from 64 to 126. The value 127 indicates a broadcast.

Related Commands

Command	Description
debug isdn event	Displays ISDN events occurring on the user side (on the router) of the ISDN interface.
debug isdn q931	Displays information about call setup and teardown of ISDN network connections (Layer 3) between the local router (user side) and the network.
service timestamps debug datetime msec	Includes the time with each debug message.

debug isdn q931

To display information about call setup and teardown of ISDN network connections (Layer 3) between the local router (user side) and the network, use the **debug isdn q931** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug isdn q931 [**asn1**| **detail**| **interface** [**bri** *number*]]

no debug isdn q931

Syntax Description

asn1	(Optional) Displays ISDN Q.931 Abstract Syntax Notation number one (ASN.1) details.
detail	(Optional) Displays ISDN Q.931 packet details.
interface	(Optional) Specifies an interface for debugging.
bri *number*	(Optional) Specifies the BRI interface and selects the interface number. Valid values are from 0 to 6.

Command Modes

Privileged EXEC

Command History

Release	Modification
10.0	The **debug isdn** command was introduced.
12.3(11)T	This command was enhanced to display the contents of the Facility Information Element (IE) in textual format.
12.3(14)T	The **asn1**, **detail**, **interface**, and **bri** *number* keywords and argument were added.
12.4(6)T	This command was enhanced to display reports about SAPI 0 procedures that accept X.25 calls on the BRI D channel.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

The ISDN network layer interface provided by the router conforms to the user interface specification defined by ITU-T recommendation Q.931 and to other specifications (for example, switch type VN4). The router only tracks activities that occur on the user side, not on the network side, of the network connection. The **debug isdn q931** command output is limited to commands and responses exchanged during peer-to-peer communication carried over the D channel. This debug information does not include data sent over B channels,

which are also part of the router's ISDN interface. The peers (network layers) communicate with each other via an ISDN switch over the D channel.

A router can be the calling or the called party of the ISDN Q.931 network connection call setup and teardown procedures. If the router is the calling party, the command displays information about an outgoing call. If the router is the called party, the command displays information about an incoming call.

This command decodes parameters of the Facility IE and displays them as text, with parameter values as they are applicable and relevant to the operation. In addition, the ASN.1 encoded Notification structure of the Notification-Indicator IE is also decoded.

You can use the **debug isdn q931** command with the **debug isdn event** and the **debug isdn q921** commands at the same time. The displays will be intermingled. Use the **service timestamps debug datetime msec** global configuration command to include the time with each message.

Examples

The following is sample output from the **debug isdn q931** command of a call setup procedure for an outgoing call:

```
Router# debug isdn q931
TX -> SETUP pd = 8 callref = 0x04
 Bearer Capability i = 0x8890
 Channel ID i = 0x83
 Called Party Number i = 0x80, '415555121202'
RX <- CALL_PROC pd = 8 callref = 0x84
 Channel ID i = 0x89
RX <- CONNECT pd = 8 callref = 0x84
TX -> CONNECT_ACK pd = 8 callref = 0x04....
Success rate is 0 percent (0/5)
```

The following is sample output from the **debug isdn q931** command of a call setup procedure for an incoming call:

```
Router# debug isdn q931
RX <- SETUP pd = 8 callref = 0x06
 Bearer Capability i = 0x8890
 Channel ID i = 0x89
 Calling Party Number i = 0x0083, '81012345678902'
TX -> CONNECT pd = 8 callref = 0x86
RX <- CONNECT_ACK pd = 8 callref = 0x06
```

The following is sample output from the **debug isdn q931** command that shows the contents of the Facility IE. The following example uses the supplementary service Malicious Call Identification (MCID). In this service, the router sends out the Facility IE.

```
Router# debug isdn q931
Sep 20 04:09:38.335 UTC: ISDN Se7/1:23 Q931: TX -> DISCONNECT pd = 8  callref = 0x0007
        Cause i = 0x8290 - Normal call clearing
        Facility i = 0x91A106020107020103
                Protocol Profile = Remote Operations Protocol
                0xA106020107020103
                Component = Invoke component
                        Invoke Id = 7 <MCID>
                        Operation = MCIDRequest
```

The following is sample output from the **debug isdn q931** command of a call teardown procedure from the network:

```
Router# debug isdn q931
RX <- DISCONNECT pd = 8 callref = 0x84
 Cause i = 0x8790
 Looking Shift to Codeset 6
 Codeset 6 IE 0x1 1 0x82 '10'
TX -> RELEASE pd = 8 callref = 0x04
```

```
 Cause i = 0x8090
RX <- RELEASE_COMP pd = 8 callref = 0x84
```

The following example shows how to turn on the **debug isdn q931 asn1** capability and how to use the **show debug** command to display the results of the debug:

```
Router# debug isdn q931 asn1
debug isdn asn1 is ON.
Router# show debug

The following ISDN debugs are enabled on all DSLs:
debug isdn error is ON.
debug isdn event is ON.
debug isdn q931 is ON.
debug isdn asn1 is ON.
DEBUGS with ASN1 enabled:
ice call = 0x1
00:08:49: Sub Msg = CDAPI_MSG_SUBTYPE_TBCT_REQ
00:08:49: Call Type = VOICE
00:08:49: B Channel = 0
00:08:49: Cause = 0
00:08:49: ISDN ASN1: isdnAsn1Component
00:08:49: ISDN ASN1: isdnAsn1Invoke
00:08:49: ISDN ASN1: isdnAsn1InvTBCT
00:08:49: ISDN ASN1: op Invoke TBCT
00:08:49: ISDN Se0:23 Q931: TX -> FACILITY pd = 8 callref = 0x8001
Facility i = 0x91A11102010506072A8648CE1500083003020101
*Jun 15 06:27:51.547: %ISDN-6-CONNECT: Interface Serial0:0 is now connected to 1 11111
00:08:51: ISDN Se0:23 Q931: RX <- FACILITY pd = 8 callref = 0x01
Facility i = 0x91A203020105A113020101800105060 72A8648CE15000A81020164
00:08:51: ISDN ASN1: isdnAsn1Component
00:08:51: ISDN ASN1: isdnAsn1Res
00:08:51: ISDN ASN1: isdnAsn1ResTbct
```

The table below describes the significant fields shown in the displays, in alphabetical order.

Table 12: debug isdn q931 Field Descriptions

Field	Description
Bearer Capability	Indicates the requested bearer service to be provided by the network.
CALL_PROC	Indicates the CALL PROCEEDING message; the requested call setup has begun, and no more call setup information will be accepted.
Called Party Number	Identifies the called party. This field is present only in outgoing SETUP messages. Note that this field can be replaced by the Keypad facility field. This field uses the IA5 character set.
Calling Party Number	Identifies the origin of the call. This field is present only in incoming SETUP messages. This field uses the IA5 character set.

Field	Description
callref	Indicates the call reference number in hexadecimal notation. The value of this field indicates the number of calls made from either the router (outgoing calls) or the network (incoming calls). Note that the originator of the SETUP message sets the high-order bit of the call reference number to 0. The destination of the connection sets the high-order bit to 1 in subsequent call control messages, such as the CONNECT message. For example, callref = 0x04 in the request becomes callref = 0x84 in the response.
Cause	Indicates the cause of the disconnect.
Channel ID	Indicates the channel identifier. The value 83 indicates any channel, 89 indicates the B1 channel, and 8A indicates the B2 channel. For more information about the channel identifier, see ITU-T Recommendation Q.931.
Codeset 6 IE 0x1 i = 0x82, '10'	Indicates charging information. This information is specific to the NTT switch type and may not be sent by other switch types.
CONNECT	Indicates that the called user has accepted the call.
CONNECT_ACK	Indicates that the calling user acknowledges the called user's acceptance of the call.
DISCONNECT	Indicates either that the user side has requested the network to clear an end-to-end connection or that the network has cleared the end-to-end connection.
i =	Indicates the information element identifier. The value depends on the field with which the identifier is associated. See the ITU-T Q.931 specification for details about the possible values associated with each field for which this identifier is relevant.
Looking Shift to Codeset 6	Indicates that the next information elements will be interpreted according to information element identifiers assigned in codeset 6. Codeset 6 means that the information elements are specific to the local network.

Field	Description
pd	Indicates the protocol discriminator that distinguishes messages for call control over the user-network ISDN interface from other ITU-T-defined messages, including other Q.931 messages. The protocol discriminator is 8 for call control messages, such as SETUP. For basic-1tr6, the protocol discriminator is 65.
Protocol Profile	Remote operations protocol, which contains networking extensions for other services. This profile determines which protocol should be used to decode the rest of a Facility IE message. A Facility IE can contain multiple components. Each component displays a hexadecimal code followed by the code contents in text. In the example that included encoded ISDN Facility IE message output, 0xA106020107020103 is the hexadecimal code and represents the Facility IE Component, Invoke Id, and Operation. The Operation portion of the IE corresponds to the supplementary service that the component represents.
RELEASE	Indicates that the sending equipment will release the channel and call reference. The recipient of this message should prepare to release the call reference and channel.
RELEASE_COMP	Indicates that the sending equipment has received a RELEASE message and has now released the call reference and channel.
RX <-	Indicates that this message is being received by the user side of the ISDN interface from the network side.
SETUP	Indicates that the SETUP message type has been sent to initiate call establishment between peer network layers. This message can be sent from either the local router or the network.
TX ->	Indicates that this message is being sent from the local router (user side) to the network side of the ISDN interface.

Text in bold in the following example indicates the acceptance of an incoming X.25 call on the ISDN D channel, per ITU Q.931 SAPI value 0 procedures:

```
Router# debug isdn q931
```

```
*Sep 28 12:34:29.739: ISDN BR1/1 Q931: RX <- SETUP pd = 8  callref = 0x5C (re-assembled)
        Bearer Capability i = 0x88C0C2E6
                Standard = CCITT
                Transfer Capability = Unrestricted Digital
                Transfer Mode = Packet
                Transfer Rate = Packet - not specified
                User Info L2 Protocol = Recommendation Q921/I.441
                User Info L3 Protocol = Recommendation X.25, Packet Layer

        Channel ID i = 0x8C
                Exclusive, No B-channel

        Information Rate i = 0x8888
        Packet Layer Binary Params i = 0x80
        Packet Layer Window Size i = 0x8282
        Packet Size i = 0x8888
        Calling Party Number i = 0x0083, '144014384106'
                Plan:Unknown, Type:Unknown
        User-User i = 0x02CC000000
```

The command output is intermingled with information from the **debug isdn events** command; see the description for the **debug isdn events** command to understand significant fields displayed in this report.

Related Commands

Command	Description
debug isdn events	Displays ISDN events occurring on the router (user side) of the ISDN interface.
debug isdn q921	Displays Layer 2 access procedures that are taking place at the router on the D channel of the ISDN interface.
service timestamps	Configure a time stamp on debugging or system logging messages.

debug isdn tgrm

To view ISDN trunk group resource manager information, use the **debug isdn tgrm** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug isdn tgrm

no debug isdn tgrm

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(11)T	This command was introduced.

Usage Guidelines

Disable console logging and use buffered logging before using the **debug isdn tgrm** command. Using the **debug isdn tgrm** command generates a large volume of debugs, which can affect router performance.

Examples

Sample output from the **debug isdn tgrm** command is shown below.

The output shows that the channel used (bchan) is 1, service state is 0 (in-service), call_state is 2 (busy), "false busy" is 0, and DSL is 2. The output also shows that the B channel is 1, the channel is available, and the call state is transitioned from 0 (idle) to 2 (busy).

The last two lines of output shows that bchan is 1, call state is 1 (busy), call type is 2 (voice), and call direction is 1 (incoming).

```
00:26:31:ISDN:get_tgrm_avail_state:idb 0x64229380 bchan 1 service_state 0 call_state 2 false
 busy 0x0 dsl 2
00:26:31:ISDN:update_tgrm_call_status:idb 0x64229380 bchan 1 availability state 1 call
state(prev,new) (0,2), dsl 2
00:26:31:ISDN:Calling TGRM with tgrm_call_isdn_update:idb 0x64229380 bchan 1 call state 1
call type 2 call dir 1
```

The table below provides an alphabetical listing of the fields shown in the **debug isdn tgrm** command output and a description of each field.

Table 13: debug isdn tgrm Field Descriptions

Field	Description
availability state	Indicates whether the channel is available: 0 = Not available 1 = Available
bchan	Bearer channel used for this call.
call dir	Direction of the call: 0 = Incoming 1 = Outgoing
call_state	State of the call. It has different values depending on whether it is from ISDN perspective or TGRM perspective. When printed from get_tgrm_avail_state(), it is the state value from ISDN perspective: 0 = Idle 1 = Negotiate 2 = Busy 3 = Reserved 4 = Restart pending 5 = Maintenance pend 6 = Reassigned When printed from tgrm_call_isdn_update(), it is the state value from TGRM perspective: 0 = Idle 1 = Busy 2 = Pending 3 = Reject
call state (prev, new)	Indicates the state transition of the call. The state values are as shown in call_state from the ISDN perspective.
call type	Type of call: 0 = Invalid 1 = Data 2 = Voice 3 = Modem 4 = None
dsl	Internal interface identifier.
false busy	Bit map of all the channels on the interface indicating their soft busy status.
idb	Address of the interface descriptor block (IDB) for the interface.
service_state	Service state: 0 = In-service 1 = Maintenance 2 = Out of service

Related Commands

Command	Description
show trunk group	Displays the configuration of the trunk group.

Command	Description
translation-profile (voice service POTS)	Assigns a translation profile to the interface.
trunk-group (interface)	Assigns a trunk group to the interface.

debug isis adj packets

To display information on all a djacency-related activity such as hello packets sent and received and Intermediate System-to-Intermediate System (IS-IS) adjacencies going up and down, use the **debug isis adj packets** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug isis adj packets [*interface*]

no debug isis adj packets [*interface*]

Syntax Description

interface	(Optional) Interface or subinterface name.

Command Modes

Privileged EXEC

Examples

The following is sample output from the **debug isis adj packets** command:

```
Router# debug isis adj packets
ISIS-Adj: Rec L1 IIH from 0000.0c00.40af (Ethernet0), cir type 3, cir id BBBB.BBBB.BBBB.01
ISIS-Adj: Rec L2 IIH from 0000.0c00.40af (Ethernet0), cir type 3, cir id BBBB.BBBB.BBBB.01
ISIS-Adj: Rec L1 IIH from 0000.0c00.0c36 (Ethernet1), cir type 3, cir id CCCC.CCCC.CCCC.03
ISIS-Adj: Area mismatch, level 1 IIH on Ethernet1
ISIS-Adj: Sending L1 IIH on Ethernet1
ISIS-Adj: Sending L2 IIH on Ethernet1
ISIS-Adj: Rec L2 IIH from 0000.0c00.0c36 (Ethernet1), cir type 3, cir id BBBB.BBBB.BBBB.03
```

The following line indicates that the router received an IS-IS hello packet (IIH) on Ethernet interface 0 from the Level 1 router (L1) at MAC address 0000.0c00.40af. The circuit type is the interface type:

```
1--Level 1 only; 2--Level 2 only; 3--Level 1/2
```

The circuit ID is what the neighbor interprets as the designated router for the interface.

```
ISIS-Adj: Rec L1 IIH from 0000.0c00.40af (Ethernet0), cir type 3, cir id BBBB.BBBB.BBBB.01
```

The following line indicates that the router (configured as a Level 1 router) received on Ethernet interface 1 is an IS-IS hello packet from a Level 1 router in another area, thereby declaring an area mismatch:

```
ISIS-Adj: Area mismatch, level 1 IIH on Ethernet1
```

The following lines indicates that the router (configured as a Level 1/Level 2 router) sent on Ethernet interface 1 is a Level 1 IS-IS hello packet, and then a Level 2 IS-IS packet:

```
ISIS-Adj: Sending L1 IIH on Ethernet1
ISIS-Adj: Sending L2 IIH on Ethernet1
```

debug isis authentication

To enable debugging of Intermediate System-to-Intermediate System (IS-IS) authentication, use the **debug isis authentication**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug isis authentication information

no debug isis authentication information

Syntax Description

information	Required keyword that specifies IS-IS authentication information.

Command Default

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(21)ST	This command was introduced.
12.2(13)T	This command was integrated into Cisco IOS Release 12.2(13)T.
12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.

Examples

The following is sample output from the **debug isis authentication**command with the **information** keyword:

```
Router# debug isis authentication information
3d03h:ISIS-AuthInfo:No auth TLV found in received packet
3d03h:ISIS-AuthInfo:No auth TLV found in received packet
```

The sample output indicates that the router has been running for 3 days and 3 hours. Debugging output is about IS-IS authentication information. The local router is configured for authentication, but it received a packet that does not contain authentication data; the remote router does not have authentication configured.

debug isis ipv6 rib

To display debugging information for Integrated Intermediate System-to-Intermediate System (IS-IS) IPv6 Version 6 routes in the global or local Routing Information Base (RIB), use the **debug isis rib**command in privileged EXEC mode. To disable the debugging of IS-IS IPv6 routes, use the **no** form of this command.

debug isis ipv6 rib [**global**| **local** [**access-list-number**| **terse**]]

no debug isis ipv6 rib [**global**| **local**]

Syntax Description

global	(Optional) Displays debugging information for IS-IS IP Version 4 routes in the global RIB.
local	(Optional) Displays debugging information for IS-IS IP Version 4 routes in the IS-IS local RIB.
access-list-number	(Optional) Number of an access list. This is a decimal number from 100 to 199 or from 2000 to 2699.
terse	(Optional) Will not display debug information if the IS-IS IP Version 4 IS-IS local RIB has not changed.

Command Default

Debugging of IS-IS IPv6 routes is disabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
Cisco IOS XE Release 3.6S	This command was introduced.

Usage Guidelines

Examples

The following is sample output from the **debug isis ipv6 rib**command shows shows an IPv6 prefix tag.

The table below describes the significant fields shown in the display.

Table 14: debug isis ipv6 rib Field Descriptions

Field	Description

Related Commands

Command	Description
isis ipv6 tag	Configures an administrative tag value to be associated with an IPv6 address prefix.

debug isis mpls traffic-eng advertisements

To print information about traffic engineering advertisements in Intermediate System-to-Intermediate System (IS-IS) link-state advertisement (LSA) messages, use the **debug isis mpls traffic-eng advertisements** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug isis mpls traffic-eng advertisements

no debug isis mpls traffic-eng advertisements

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(5)ST	This command was introduced.
12.1(3)T	This command was integrated into Cisco IOS Release 12.1(3)T.
12.0(22)S	This command was integrated into Cisco IOS Release 12.0(22)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Examples

In the following example, information about traffic engineering advertisements is printed in IS-IS LSA messages:

```
Router# debug isis mpls traffic-eng advertisements
System ID:Router1.00
  Router ID:10.106.0.6
  Link Count:1
    Link[1]
      Neighbor System ID:Router2.00 (P2P link)
      Interface IP address:10.42.0.6
      Neighbor IP Address:10.42.0.10
      Admin. Weight:10
      Physical BW:155520000 bits/sec
      Reservable BW:5000000 bits/sec
      BW unreserved[0]:2000000 bits/sec, BW unreserved[1]:100000 bits/sec
      BW unreserved[2]:100000 bits/sec, BW unreserved[3]:100000 bits/sec
      BW unreserved[4]:100000 bits/sec, BW unreserved[5]:100000 bits/sec
      BW unreserved[6]:100000 bits/sec, BW unreserved[7]:0 bits/sec
      Affinity Bits:0x00000000
```

The table below describes the significant fields shown in the display.

Table 15: debug isis mpls traffic-eng advertisements Field Descriptions

Field	Description
System ID	Identification value for the local system in the area.
Router ID	Multiprotocol Label Switching traffic engineering router ID.
Link Count	Number of links that MPLS traffic engineering advertised.
Neighbor System ID	Identification value for the remote system in an area.
Interface IP address	IPv4 address of the interface.
Neighbor IP Address	IPv4 address of the neighbor.
Admin. Weight	Administrative weight associated with this link.
Physical BW	Bandwidth capacity of the link (in bits per second).
Reservable BW	Amount of reservable bandwidth on this link.
BW unreserved	Amount of bandwidth that is available for reservation.
Affinity Bits	Attribute flags of the link that are being flooded.

debug isis mpls traffic-eng events

To print information about traffic engineering-related Intermediate System-to-Intermediate System (IS-IS) events, use the **debug isis mpls traffic-eng events** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug isis mpls traffic-eng events

no debug isis mpls traffic-eng events

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(5)ST	This command was introduced.
12.1(3)T	This command was integrated into Cisco IOS Release 12.1(3)T.
12.0(22)S	This command was integrated into Cisco IOS Release 12.0(22)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Examples

In the following example, information is printed about traffic engineering-related IS-IS events:

```
Router# debug isis mpls traffic-eng events
ISIS-RRR:Send MPLS TE Et4/0/1 Router1.02 adjacency down:address 0.0.0.0
ISIS-RRR:Found interface address 10.1.0.6 Router1.02, building subtlv... 58 bytes
ISIS-RRR:Found interface address 10.42.0.6 Router2.00, building subtlv... 64 bytes
ISIS-RRR:Interface address 0.0.0.0 Router1.00 not found, not building subtlv
ISIS-RRR:LSP Router1.02 changed from 0x606BCD30
ISIS-RRR:Mark LSP Router1.02 changed because TLV contents different, code 16
ISIS-RRR:Received 1 MPLS TE links flood info for system id Router1.00
```

debug isis nsf

To display information about the Intermediate System-to-Intermediate System (IS-IS) state during a Cisco nonstop forwarding (NSF) restart, use the **d ebug isis nsf** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug isis nsf [detail]

no debug isis nsf [detail]

Syntax Description

detail	(Optional) Provides detailed debugging information.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(22)S	This command was introduced.
12.2(18)S	This command was integrated into Cisco IOS Release 12.2(18)S.
12.2(20)S	Support for the Cisco 7304 router was added.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.

Usage Guidelines

Use the **debug isis nsf**command to display basic information about the IS-IS state during an NSF restart. Use the **debug isis nsf detail**command to display additional IS-IS state detail during an NSF restart.

Examples

The following example displays IS-IS state information during an NSF restart:

```
router# debug isis nsf
IS-IS NSF events debugging is on
```

The following example displays detailed IS-IS state information during an NSF restart:

```
router# debug isis nsf detail
IS-IS NSF events (detailed) debugging is on
router#
Jan 24 20:04:54.090:%CLNS-5-ADJCHANGE:ISIS:Adjacency to gsr1 (GigabitEthernet2/0/0) Up,
Standby adjacency
Jan 24 20:04:54.090:ISIS-NSF:ADJ:000C.0000.0000 (Gi2/0/0), type 8/1, cnt 0/1, ht 10 (NEW)
Jan 24 20:04:54.142:ISIS-NSF:Rcv LSP - L2 000B.0000.0000.00-00, seq 251, csum B0DC, ht 120,
 len 123 (local)
```

```
Jan 24 20:04:55.510:ISIS-NSF:Rcv LSP - L1 000B.0000.0000.00-00, seq 23E, csum D20D, ht 120,
 len 100 (local)
Jan 24 20:04:56.494:ISIS-NSF:ADJ:000C.0000.0000 (Gi2/0/0), type 8/0, cnt 0/1, ht 30
Jan 24 20:04:56.502:ISIS-NSF:Rcv LSP - L1 000B.0000.0000.01-00, seq 21C, csum 413, ht 120,
 len 58 (local)
Jan 24 20:04:58.230:ISIS-NSF:Rcv LSP - L2 000C.0000.0000.00-00, seq 11A, csum E197, ht 1194,
 len 88 (Gi2/0/0)
Jan 24 20:05:00.554:ISIS-NSF:Rcv LSP - L1 000B.0000.0000.00-00, seq 23F, csum 1527, ht 120,
 len 111 (local)
```

Related Commands

Command	Description
nsf (IS-IS)	Configures NSF operations for IS-IS.
nsf interface wait	Specifies how long an NSF restart will wait for all interfaces with IS-IS adjacencies to come up before completing the restart.
nsf interval	Specifies the minimum time between NSF restart attempts.
nsf t3	Specifies the methodology used to determine how long IETF NSF will wait for the LSP database to synchronize before generating overloaded link state information for itself and flooding that information out to its neighbors.
show clns neighbors	Displays both ES and IS neighbors.
show isis nsf	Displays current state information regarding IS-IS NSF.

debug isis rib

To display debugging information for Integrated Intermediate System-to-Intermediate System (IS-IS) IP Version 4 routes in the global or local Routing Information Base (RIB), use the **debug isis rib**command in privileged EXEC mode. To disable the debugging of IS-IS IP Version 4 routes, use the **no** form of this command.

debug isis rib [**global**| **local** [**access-list-number**| **terse**]]

no debug isis rib [**global**| **local**]

Syntax Description

global	(Optional) Displays debugging information for IS-IS IP Version 4 routes in the global RIB.
local	(Optional) Displays debugging information for IS-IS IP Version 4 routes in the IS-IS local RIB.
access-list-number	(Optional) Number of an access list. This is a decimal number from 100 to 199 or from 2000 to 2699.
terse	(Optional) Will not display debug information if the IS-IS IP Version 4 IS-IS local RIB has not changed.

Command Default

Debugging of IS-IS IP Version 4 routes is disabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(26)S	This command was introduced.
12.3(4)T	This command was integrated into Cisco IOS Release 12.3(4)T.
12.2(25)S	This command was integrated into Cisco IOS Release 12.2(25)S.
12.2(18)SXE	This command was integrated into Cisco IOS Release 12.2(18)SXE.
12.2(27)SBC	This command was integrated into Cisco IOS Release 12.2(27)SBC.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

Use the **debug isis rib** command to verify if an IP prefix has been installed or removed. To monitor updates from the IS-IS database to the IS-IS local RIB, use the **local** keyword, and to monitor updates from the IS-IS database to the global RIB, use the **global** keyword.

It is highly recommended that you limit the debugging output to information specific to the IP prefix that is associated with a specific access list by entering the *accest-list-number* argument.

Examples

The following is sample output from the **debug isis rib**command after the **ip route priority high**command was used to give high priority to IS-IS IP prefixes for the configured access list access-list1. The debug output shows that the route 10.1.1.0/24 has been removed from the IS-IS local RIB.

```
Router# show running-config| include access-list 1
accest-list 1 permit 10.1.1.0 0.0.0.255
! access-list 1 is configured
Router# debug isis rib local terse 1
00:07:07: ISIS-LR: 10.1.1.0/24 aged out in LSP[10/(7->8)]
! The route 10.1.1.0/24 is removed from the IS-IS local RIB LSP[10/(7->8)].
00:07:07: ISIS-LR: rem path: [115/80/20] via 10.2.2.2(Et2) from 10.22.22.22 tg 0 LSP[10/7]
 from active chain (add to deleted chain)
!The remote path [115/80/20] is removed from the active chain.
00:07:07: ISIS-LR: Enqueued to updateQ[2] for 10.1.1.0/24
!Q[2] is marked to be the update
00:07:07: ISIS-LR: rem path: [115/80/20] via 10.2.2.2(Et2) from 10.22.22.22 tg 0 LSP[10/7]
 from deleted chain
00:07:07: ISIS-LR: Rem RT 10.1.1.0/24
!The remote route [115/80/20] is removed from the deleted chain
```

The table below describes the significant fields shown in the display.

Table 16: debug isis rib Field Descriptions

Field	Description
ISIS-LR	IS-IS local route debugger.
10.1.1.0/24	IP prefix.
rem path:	Indicates the removal or insertion of a routing path--in this instance, it is a removal.
[115/80/20]	Administrative instance/type/metric for the routing path that has been removed or inserted.
via 10.2.2.2(Et2)	IP address of the next hop of the router, in this instance, Ethernet2.
from 10.22.22.22	IP address to advertise the route path.
tg 0	Priority of the IP prefix. All prefixes have a tag 0 priority unless otherwise configured.

Related Commands

Command	Description
ip route priority high	Assigns a high priority to an IS-IS IP prefix.
show isis rib	Displays paths for routes in the IP Version 4 IS-IS local RIB.

debug isis rib redistribution

To debug the events that update the Intermediate System-to-Intermediate System (IS-IS) redistribution cache, use the **debug isis rib redistribution**command in privileged EXEC mode. To turn off debugging, use the **no** form of this command.

debug isis rib redistribution [**level-1**| **level-2**] [*access-list*]

no debug isis rib redistribution [**level-1**| **level-2**] [*access-list*]

Syntax Description

level-1	(Optional) Displays debug information for level 1 redistribution cache.
level-2	(Optional) Displays debug information for level 2 redistribution cache.
access-list	(Optional) An access list number from 1 to 199 or from 1300 to 2699.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(27)S	This command was introduced.
12.3(7)T	This command was integrated into Cisco IOS Release 12.3(7)T.
12.2(25)S	This command was integrated into Cisco IOS Release 12.2(25)S.
12.2(18)SXE	This command was integrated into Cisco IOS Release 12.2(18)SXE.
12.2(27)SBC	This command was integrated into Cisco IOS Release 12.2(27)SBC.

Usage Guidelines

We recommend that you use this command only when a Cisco Technical Assistance Center representative requests you to do so to gather information for a troubleshooting purpose.

Examples

In the following example, the **debug isis rib redistribution** command is used to display information about events that update the IS-IS redistribution cache. The output is self-explanatory.

```
Router# debug isis rib redistribution level-1 123
IS-IS IPv4 redistribution RIB debugging is on for access list 123 for L1
Router# router isis
Router(config-router)# redistribute connected level-1
```

```
Router(config)# access-list 123 permit ip 10.0.0.0 0.255.255.255 any
Router(config)# interface Loopback123
Router(config-if)# ip address 10.123.123.3 255.255.255.255
Nov 25 00:33:46.532: ISIS-RR: 10.123.123.3/32: Up event, from 0x607CAF60
Nov 25 00:33:46.532: ISIS-RR:   looking at L1 redist RIB
Nov 25 00:33:46.532: ISIS-RR:     redistributed to ISIS
Nov 25 00:33:46.532: ISIS-RR:   added    10.123.123.3/32 to L1 redist RIB: [Connected/0]
tag 0 external
Nov 25 00:33:47.532: ISIS-RR: Scanning L1 redist RIB
Nov 25 00:33:47.532: ISIS-RR:   adv 10.123.123.3/32 as L1 redist route
Nov 25 00:33:47.532: ISIS-RR: End of scanningL1 redist RIB
```

The following line indicates that the connected route 10.123.123.3/32 was added to the IS-IS level 1 local redistribution cache with cost 0, metric type external, and administrative tag of 0:

```
Nov 25 00:33:46.532: ISIS-RR:   added    10.123.123.3/32 to L1 redist RIB: [Connected/0]
tag 0 external
```

The following line indicates that the redistributed route 10.123.123.3/32 was advertised in an IS-IS link-state packet (LSP) as a level 1 redistributed route:

```
Nov 25 00:33:47.532: ISIS-RR:   adv 10.123.123.3/32 as L1 redist rout
```

Related Commands

Command	Description
clear isis rib redistribution	Clears some or all prefixes in the local redistribution cache.
show isis rib redistribution	Displays the prefixes in the IS-IS redistribution cache.

debug isis spf statistics

To display statistical information about building routes between intermediate systems (ISs), use the **debug isis spf statistics** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug isis spf statistics

no debug isis spf statistics

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

The Intermediate System-to-Intermediate System (IS-IS) Interdomain Routing Protocol (IDRP) provides routing between ISs by flooding the network with link-state information. IS-IS provides routing at two levels, intra-area (Level 1) and intra-domain (Level 2). Level 1 routing allows Level 1 ISs to communicate with other Level 1 ISs in the same area. Level 2 routing allows Level 2 ISs to build an interdomain backbone between Level 1 areas by traversing only Level 2 ISs. Level 1 ISs only need to know the path to the nearest Level 2 IS in order to take advantage of the interdomain backbone created by the Level 2 ISs.

The IS-IS protocol uses the shortest-path first (SPF) routing algorithm to build Level 1 and Level 2 routes. The **debug isis spf statistics** command provides information for determining the time required to place a Level 1 IS or Level 2 IS on the shortest path tree (SPT) using the IS-IS protocol.

Note The SPF algorithm is also called the Dijkstra algorithm, after the creator of the algorithm.

Examples

The following is sample output from the **debug isis spf statistics** command:

```
Router# debug isis spf statistics
ISIS-Stats: Compute L1 SPT, Timestamp 2780.328 seconds
ISIS-Stats: Complete L1 SPT, Compute time 0.004, 1 nodes on SPT
ISIS-Stats: Compute L2 SPT, Timestamp 2780.3336 seconds
ISIS-Stats: Complete L2 SPT, Compute time 0.056, 12 nodes on SPT
```

The table below describes the significant fields shown in the display.

Table 17: debug isis spf statistics Field Descriptions

Field	Description
Compute L1 SPT	Indicates that Level 1 ISs are to be added to a Level 1 area.

Field	Description
Timestamp	Indicates the time at which the SPF algorithm was applied. The time is expressed as the number of seconds elapsed since the system was up and configured.
Complete L1 SPT	Indicates that the algorithm has completed for Level 1 routing.
Compute time	Indicates the time required to place the ISs on the SPT.
nodes on SPT	Indicates the number of ISs that have been added.
Compute L2 SPT	Indicates that Level 2 ISs are to be added to the domain.
Complete L2 SPT	Indicates that the algorithm has completed for Level 2 routing.

The following lines show the statistical information available for Level 1 ISs:

```
ISIS-Stats: Compute L1 SPT, Timestamp 2780.328 seconds
ISIS-Stats: Complete L1 SPT, Compute time 0.004, 1 nodes on SPT
```

The output indicates that the SPF algorithm was applied 2780.328 seconds after the system was up and configured. Given the existing intra-area topology, 4 milliseconds were required to place one Level 1 IS on the SPT.

The following lines show the statistical information available for Level 2 ISs:

```
ISIS-Stats: Compute L2 SPT, Timestamp 2780.3336 seconds
ISIS-Stats: Complete L2 SPT, Compute time 0.056, 12 nodes on SPT
```

This output indicates that the SPF algorithm was applied 2780.3336 seconds after the system was up and configured. Given the existing intradomain topology, 56 milliseconds were required to place 12 Level 2 ISs on the SPT.

debug isis spf-events

To display a log of significant events during an Intermediate System-to-Intermediate System (IS-IS) shortest-path first (SPF) computation, use the **debug isis spf-events**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug isis spf-events

no debug isis spf-events

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0	This command was introduced.
12.2(15)T	Support for IPv6 was added.
12.2(18)S	Support for IPv6 was added.
12.0(26)S	Support for IPv6 was added.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
Cisco IOS XE Release 2.6	This command was introduced on Cisco ASR 1000 series routers.

Usage Guidelines

This command displays information about significant events that occur during SPF-related processing.

Examples

The following example displays significant events during an IS-IS SPF computation:

```
Router# debug isis spf-events
ISIS-Spf:  Compute L2 IPv6 SPT
ISIS-Spf: Move 0000.0000.1111.00-00 to PATHS, metric 0
ISIS-Spf: Add 0000.0000.2222.01-00 to TENT, metric 10
ISIS-Spf: Move 0000.0000.2222.01-00 to PATHS, metric 10
ISIS-Spf: considering adj to 0000.0000.2222 (Ethernet3/1) metric 10, level 2, circuit 3,
adj 3
ISIS-Spf:   (accepted)
ISIS-Spf: Add 0000.0000.2222.00-00 to TENT, metric 10
ISIS-Spf:   Next hop 0000.0000.2222 (Ethernet3/1)
ISIS-Spf: Move 0000.0000.2222.00-00 to PATHS, metric 10
ISIS-Spf: Add 0000.0000.2222.02-00 to TENT, metric 20
```

```
ISIS-Spf:   Next hop 0000.0000.2222 (Ethernet3/1)
ISIS-Spf: Move 0000.0000.2222.02-00 to PATHS, metric 20
ISIS-Spf: Add 0000.0000.3333.00-00 to TENT, metric 20
ISIS-Spf:   Next hop 0000.0000.2222 (Ethernet3/1)
ISIS-Spf: Move 0000.0000.3333.00-00 to PATHS, metric 20
```

debug isis update-packets

To display various sequence number protocol data units (PDUs) and link-state packets that are detected by a router, use the **debug isis update-packets** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug isis update-packets

no debug isis update-packets

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Examples

This router has been configured for IS-IS routing. The following is sample output from thee **debug isis update-packets** command:

```
Router# debug isis update-packets
ISIS-Update: Sending L1 CSNP on Ethernet0
ISIS-Update: Sending L2 CSNP on Ethernet0
ISIS-Update: Updating L2 LSP
ISIS-Update: Delete link 888.8800.0181.00 from L2 LSP 1600.8906.4022.00-00, seq E
ISIS-Update: Updating L1 LSP
ISIS-Update: Sending L1 CSNP on Ethernet0
ISIS-Update: Sending L2 CSNP on Ethernet0
ISIS-Update: Add link 8888.8800.0181.00 to L2 LSP 1600.8906.4022.00-00, new seq 10,
 len 91
ISIS-Update: Sending L2 LSP 1600.8906.4022.00-00, seq 10, ht 1198 on Tunnel0
ISIS-Update: Sending L2 CSNP on Tunnel0
ISIS-Update: Updating L2 LSP
ISIS-Update: Rate limiting L2 LSP 1600.8906.4022.00-00, seq 11 (Tunnel0)
ISIS-Update: Updating L1 LSP
ISIS-Update: Rec L2 LSP 888.8800.0181.00.00-00 (Tunnel0)
ISIS-Update: PSNP entry 1600.8906.4022.00-00, seq 10, ht 1196
```

The following lines indicate that the router has sent a periodic Level 1 and Level 2 complete sequence number PDU on Ethernet interface 0:

```
ISIS-Update: Sending L1 CSNP on Ethernet0
ISIS-Update: Sending L2 CSNP on Ethernet0
```

The following lines indicate that the network service access point (NSAP) identified as 8888.8800.0181.00 was deleted from the Level 2 LSP 1600.8906.4022.00-00. The sequence number associated with this LSP is 0xE.

```
ISIS-Update: Updating L2 LSP
ISIS-Update: Delete link 888.8800.0181.00 from L2 LSP 1600.8906.4022.00-00, seq E
```

The following lines indicate that the NSAP identified as 8888.8800.0181.00 was added to the Level 2 LSP 1600.8906.4022.00-00. The new sequence number associated with this LSP is 0x10.

```
ISIS-Update: Updating L1 LSP
ISIS-Update: Sending L1 CSNP on Ethernet0
ISIS-Update: Sending L2 CSNP on Ethernet0
ISIS-Update: Add link 8888.8800.0181.00 to L2 LSP 1600.8906.4022.00-00, new seq 10,
 len 91
```

The following line indicates that the router sent Level 2 LSP 1600.8906.4022.00-00 with sequence number 0x10 on tunnel 0 interface:

```
ISIS-Update: Sending L2 LSP 1600.8906.4022.00-00, seq 10, ht 1198 on Tunnel0
```

The following lines indicates that a Level 2 LSP could not be transmitted because it was recently sent:

```
ISIS-Update: Sending L2 CSNP on Tunnel0
ISIS-Update: Updating L2 LSP
ISIS-Update: Rate limiting L2 LSP 1600.8906.4022.00-00, seq 11 (Tunnel0)
```

The following lines indicate that a Level 2 partial sequence number PDU (PSNP) has been received on tunnel 0 interface:

```
ISIS-Update: Updating L1 LSP
ISIS-Update: Rec L2 PSNP from 8888.8800.0181.00 (Tunnel0)
```

The following line indicates that a Level 2 PSNP with an entry for Level 2 LSP 1600.8906.4022.00-00 has been received. This output is an acknowledgment that a previously sent LSP was received without an error.

```
ISIS-Update: PSNP entry 1600.8906.4022.00-00, seq 10, ht 1196
```

debug iua as

To display debugging messages for the IDSN User Adaptation Layer (IUA) application server (AS), use the **debug iua as** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug iua as {**user**| **state**} {**all**| **name** *as-name*}

no debug iua as

Syntax Description

user	Displays information about the use of application programming interfaces (APIs) and events between the ISDN layer and IUA.
state	Displays information about AS state transitions.
all	Enables debug for all the configured ASs.
name *as-name*	Defines the name of the AS.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(4)T	This command was introduced.
12.2(15)T	This command was integrated into Cisco IOS Release 12.2(15)T on the Cisco 2420, Cisco 2600 series, Cisco 3600 series, and Cisco 3700 series; and Cisco AS5300, Cisco AS5350, Cisco AS5400, and Cisco AS5850 network access server (NAS) platforms.

Examples

The following example shows debugging output when an ISDN backhaul connection is initially established. The output shows that state debugging is turned on for all ASs and that the AS is active.

```
Router# debug iua as state all

IUA :state debug turned ON for ALL AS
00:11:52:IUA:AS as1 number of ASPs up is 1
00:11:57:IUA:AS as1 xsition AS-Up -->  AS-Active, cause - ASP asp1
```

Related Commands

Command	Description
debug iua asp	Displays debugging messages for the IUA ASP.

debug iua asp

To display debugging messages for the IDSN User Adaptation Layer (IUA) application server process (ASP), use the **debug iua asp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug iua asp {**pak**| **peer-msg**| **sctp-sig**| **state**} {**all**| **name** *asp-name*}

no debug iua asp

Syntax Description

pak	Displays information about all packets.
peer-msg	Displays information about IUA peer-to-peer messages.
sctp-sig	Displays information about the signals being sent by the Stream Control Transmission Protocol (SCTP) layer.
state	Displays information about ASP state transition.
all	Enables debugging output for all configured ASPs.
name *asp-name*	Defines the name of the ASP.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(4)T	This command was introduced.
12.2(15)T	This command was integrated into Cisco IOS Release 12.2(15)T on the Cisco 2420, Cisco 2600 series, Cisco 3600 series, and Cisco 3700 series; and Cisco AS5300, Cisco AS5350, Cisco AS5400, and Cisco AS5850 network access server (NAS) platforms.

Examples

The following example shows debugging output when an ISDN backhaul connection is initially established. The output shows that peer message debugging is turned on for all ASPs and that the ASP is active.

```
Router# debug iua asp peer-msg all
```

```
IUA :peer message debug turned ON for ALL ASPs
Router#
00:04:58:IUA :recieved ASP_UP message on ASP asp1
00:04:58:IUA:ASP asp1 xsition ASP-Down -->  ASP-Up , cause -  rcv peer
msg
ASP-UP
00:04:58:IUA:sending ACK of type 0x304 to asp asp1
00:05:03:IUA:recv ASP_ACTIVE message for ASP asp1
00:05:03:IUA:ASP asp1 xsition ASP-Up -->  ASP-Active, cause - rcv peer
msg
ASP-Active
```

Related Commands

Command	Description
debug iua as	Displays debugging messages for the IUA AS.

debug kerberos

To display information associated with the Kerberos Authentication Subsystem, use the **debug kerberos**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug kerberos

no debug kerberos

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

Kerberos is a security system that authenticates users and services without passing a cleartext password over the network. Cisco supports Kerberos under the authentication, authorization, and accounting (AAA) security system.

Use the **debug aaa authentication** command to get a high-level view of login activity. When Kerberos is used on the router, you can use the **debug kerberos**command for more detailed debugging information.

Examples

The following is part of the sample output from the **debug aaa authentication** command for a Kerberos login attempt that failed. The information indicates that Kerberos is the authentication method used.

```
Router# debug aaa authentication
AAA/AUTHEN/START (116852612): Method=KRB5
AAA/AUTHEN (116852612): status = GETUSER
AAA/AUTHEN/CONT (116852612): continue_login
AAA/AUTHEN (116852612): status = GETUSER
AAA/AUTHEN (116852612): Method=KRB5
AAA/AUTHEN (116852612): status = GETPASS
AAA/AUTHEN/CONT (116852612): continue_login
AAA/AUTHEN (116852612): status = GETPASS
AAA/AUTHEN (116852612): Method=KRB5
AAA/AUTHEN (116852612): password incorrect
AAA/AUTHEN (116852612): status = FAIL
```

The following is sample output from the **debug kerberos** command for a login attempt that was successful. The information indicates that the router sent a request to the key distribution center (KDC) and received a valid credential.

```
Router# debug kerberos
Kerberos: Requesting TGT with expiration date of 820911631
Kerberos: Sent TGT request to KDC
Kerberos: Received TGT reply from KDC
Kerberos: Received valid credential with endtime of 820911631
```

The following is sample output from the **debug kerberos** command for a login attempt that failed. The information indicates that the router sent a request to the KDC and received a reply, but the reply did not contain a valid credential.

```
Router# debug kerberos
Kerberos: Requesting TGT with expiration date of 820911731
Kerberos: Sent TGT request to KDC
Kerberos: Received TGT reply from KDC
```

```
Kerberos: Received invalid credential.
AAA/AUTHEN (425003829): password incorrect
```

The following output shows other failure messages you might see that indicate a configuration problem. The first message indicates that the router failed to find the default Kerberos realm, therefore the process failed to build a message to send to the KDC. The second message indicates that the router failed to retrieve its own IP address. The third message indicates that the router failed to retrieve the current time. The fourth message indicates the router failed to find or create a credentials cache for a user, which is usually caused by low memory availability.

```
Router# debug kerberos
Kerberos: authentication failed when parsing name
Kerberos: authentication failed while getting my address
Kerberos: authentication failed while getting time of day
Kerberos: authentication failed while allocating credentials cache
```

Related Commands

Command	Description
debug aaa authentication	Displays information on accountable events as they occur.

debug kpml

To enable Keypad Markup Language (KPML) parser and builder debugs, use the **debug kpml** command to specify the debug option.

To disable KPML parser and builder debugs, use the **no** form of this command (you must enter one option).

debug kpml [**all**| **parser**| **builder**| **error**]

no debug kpml [**all**| **parser**| **builder**| **error**]

Syntax Description

all	Enables all kpml debug tracing.
parser	Enables kpml parser tracing.
builder	Enables kpml builder tracing.
error	Enables kpml error tracing.

Command Default

no debug kpml all

Command Modes

Privileged EXEC mode

Command History

Release	Modification
12.4(9)T	This command was introduced.

Usage Guidelines

For incoming dial peers if you configure multiple DTMF negotiation methods, the first configure value takes precedence, then the second, then the third.

For incoming dial peers, the first out-of-band negotiation method takes precedence over other DTMF negotiation methods, except when rtp-nte has precedence; in this case, sip-kpml takes precedence over other out-of-band negotiation methods.

For incoming dial peers, if both sip-kpml and rtp-nte notification mechanisms are enabled and negotiated, the gateway relies on RFC 2833 notification to receive digits and a SUBSCRIBE for KPML is not initiated.

SIP KPML support complies to the IEFT draft “draft-ietf-sipping-kpml-04.txt” with the following limitations:

- The SIP gateway always initiates SUBSCRIBE in the context of an established INVITE dialog. The gateway supports receiving SUBSCRIBE in the context of an established INVITE dialog, as well as out-of-call context requests with a leg parameter in the Event header. If the request code does not match an existing INVITE dialog, the gateway sends a NOTIFY with KPML status-code 481 and sets Subscription-State to terminated.

- The gateway does not support the Globally Routable User Agent (GRUU) requirement. The Contact header in the INVITE/200 OK message generates locally from the gateway's contact information.
- The gateway always initiates persistent subscriptions, but it receives and processes persistent and one-shot subscriptions.
- The gateway supports only single-digit reporting. There is no need for inter-digit timer support. The only regular expressions supported are those which match to a single digit. For example:
  - <regex>x</regex>--Matches to any digit 0 through 9
  - <regex>1</regex>--Matches digit 1
  - <regex>[x#*ABCD]</regex>--Matches to any digit 0 through 9, # (the pound sign), * (an asterisk), or A, B, C, or D
  - <regex>[24]</regex>--Matches digits 2 or 4
  - <regex>[2-9]</regex>--Matches on any digit 2 through 9
  - <regex>[^2-9]</regex>--Matches digits 0 or 1
- The gateway does not support long key presses. Long key presses are detected and reported as a single digit press.
- Digit suppression is not supported (*pre* tag for suppressing inband digits).
- Individual stream selection is not supported. A SUBSCRIBE request for KPML applies to all audio streams in the dialog (*stream* element and *reverse* not supported).

You can configure support only on a SIP VoIP dial peer.

Examples

The following is output from the **debug kpml** command:

```
SIP call is established.  DTMF sip-kpml was negotiated.
...
//-1/xxxxxxxxxxxx/KPML/Parser/kpml_init:
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode: encode_data=0x64E25B48
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode_context_create: chunk_size=2k, max_allowed=16k
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode_context_create: context=0x6488C0AC, mp=0x6488B89C
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_build_request:
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_build_pattern:
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_build_regex_list:
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode: malloc xml_buf=0x645E910C, length=328
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_build_request:
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_build_pattern:
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_build_regex_list:
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_build_request: length=289, buffp=0x645E9251
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode: rc=0, encoded str=<?xml version="1.0"
encoding="UTF-8"?><kpml-request xmlns="urn:ietf:params:xml:ns:kpml-request"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:ietf:params:xml:ns:kpml-request kpml-request.xsd"
version="1.0"><pattern persist="persist"><regex
tag="dtmf">[x*#ABCD]</regex></pattern></kpml-request>
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode_context_free:
kpml_encode_context_free:mem_mgr_mempool_free: mem_refcnt(6488B89C)=0 - mempool cleanup
/-1/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg:
Sent:
SUBSCRIBE sip:8888@172.18.193.250:5060 SIP/2.0
Via: SIP/2.0/UDP 172.18.193.251:5060;branch=z9hG4bKFF36
From: <sip:172.18.193.251>;tag=EA330-F6
To: <sip:8888@172.18.193.250>;tag=39497C-2EA
Call-ID: 57633F68-2BE011D6-8013D46B-B4F9B5F6@172.18.193.251
```

```
CSeq: 103 SUBSCRIBE
Max-Forwards: 70
Date: Fri, 01 Mar 2002 00:16:15 GMT
User-Agent: Cisco-SIPGateway/IOS-12.x
Event: kpml
Expires: 7200
Contact: <sip:172.18.193.251:5060>
Content-Type: application/kpml-request+xml
Content-Length: 327
<?xml version="1.0" encoding="UTF-8"?><kpml-request
xmlns="urn:ietf:params:xml:ns:kpml-request"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:ietf:params:xml:ns:kpml-request kpml-request.xsd"
version="1.0"><pattern persist="persist"><regex
tag="dtmf">[x*#ABCD]</regex></pattern></kpml-request>
/-1/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg:
Received:
SUBSCRIBE sip:172.18.193.251:5060 SIP/2.0
Via: SIP/2.0/UDP 172.18.193.250:5060;branch=z9hG4bK5FE3
From: <sip:8888@172.18.193.250>;tag=39497C-2EA
To: <sip:172.18.193.251>;tag=EA330-F6
Call-ID: 57633F68-2BE011D6-8013D46B-B4F9B5F6@172.18.193.251
CSeq: 101 SUBSCRIBE
Max-Forwards: 70
Date: Fri, 01 Mar 2002 01:02:46 GMT
User-Agent: Cisco-SIPGateway/IOS-12.x
Event: kpml
Expires: 7200
Contact: <sip:172.18.193.250:5060>
Content-Type: application/kpml-request+xml
Content-Length: 327
<?xml version="1.0" encoding="UTF-8"?><kpml-request
xmlns="urn:ietf:params:xml:ns:kpml-request"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:ietf:params:xml:ns:kpml-request kpml-request.xsd"
version="1.0"><pattern persist="persist"><regex
tag="dtmf">[x*#ABCD]</regex></pattern></kpml-request>
/-1/xxxxxxxxxxxx/KPML/Parser/kpml_init:
//-1/xxxxxxxxxxxx/KPML/Parser/kpml_decode: Parsing <?xml version="1.0"
encoding="UTF-8"?><kpml-request xmlns="urn:ietf:params:xml:ns:kpml-request"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:ietf:params:xml:ns:kpml-request kpml-request.xsd"
version="1.0"><pattern persist="persist"><regex
tag="dtmf">[x*#ABCD]</regex></pattern></kpml-request>
/-1/xxxxxxxxxxxx/KPML/Parser/kpml_request_ptbuild:
//-1/xxxxxxxxxxxx/KPML/Parser/kpml_create_new_node: creating node
par/cur/child=0x00000000/0x645E910C/0x00000000  top/child=0x645E910C/0x00000000
//-1/xxxxxxxxxxxx/KPML/Parser/kpml_pattern_ptbuild:
//-1/xxxxxxxxxxxx/KPML/Parser/kpml_create_new_node: creating node
par/cur/child=0x645E910C/0x645E91E8/0x00000000  top/child=0x645E910C/0x645E91E8
//-1/xxxxxxxxxxxx/KPML/Parser/kpml_regex_ptbuild:
//-1/xxxxxxxxxxxx/KPML/Parser/kpml_create_new_node: creating node
par/cur/child=0x645E91E8/0x645E923C/0x00000000  top/child=0x645E910C/0x645E91E8
//-1/xxxxxxxxxxxx/KPML/Parser/kpml_character_data:
buf=[x*#ABCD]</regex></pattern></kpml-request>
/-1/xxxxxxxxxxxx/KPML/Parser/kpml_regex_char_data_ptbuild: char data=[x*#ABCD]
//-1/xxxxxxxxxxxx/KPML/Parser/kpml_end_element_handler: elem name=regex
//-1/xxxxxxxxxxxx/KPML/Parser/kpml_end_element_handler: elem name=pattern
//-1/xxxxxxxxxxxx/KPML/Parser/kpml_end_element_handler: elem name=kpml-request
//-1/xxxxxxxxxxxx/KPML/Parser/kpml_pattern_ptproc:
//-1/xxxxxxxxxxxx/KPML/Parser/kpml_regex_ptproc:
//-1/xxxxxxxxxxxx/KPML/Parser/kpml_decode_context_free:
kpml_decode_context_free:mem_mgr_mempool_free: mem_refcnt(6488B89C)=0 - mempool cleanup
//-1/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg:
Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP 172.18.193.251:5060;branch=z9hG4bKFF36
From: <sip:172.18.193.251>;tag=EA330-F6
To: <sip:8888@172.18.193.250>;tag=39497C-2EA
Date: Fri, 01 Mar 2002 01:02:51 GMT
Call-ID: 57633F68-2BE011D6-8013D46B-B4F9B5F6@172.18.193.251
CSeq: 103 SUBSCRIBE
Content-Length: 0
```

```
Contact: <sip:172.18.193.250:5060>
Expires: 7200
//-1/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg:
Sent:
SIP/2.0 200 OK
Via: SIP/2.0/UDP 172.18.193.250:5060;branch=z9hG4bK5FE3
From: <sip:8888@172.18.193.250>;tag=39497C-2EA
To: <sip:172.18.193.251>;tag=EA330-F6
Date: Fri, 01 Mar 2002 00:16:24 GMT
Call-ID: 57633F68-2BE011D6-8013D46B-B4F9B5F6@172.18.193.251
CSeq: 101 SUBSCRIBE
Content-Length: 0
Contact: <sip:172.18.193.251:5060>
Expires: 7200
//-1/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg:
Sent:
NOTIFY sip:172.18.193.250:5060 SIP/2.0
Via: SIP/2.0/UDP 172.18.193.251:5060;branch=z9hG4bK101EA4
From: <sip:172.18.193.251>;tag=EA330-F6
To: <sip:8888@172.18.193.250>;tag=39497C-2EA
Call-ID: 57633F68-2BE011D6-8013D46B-B4F9B5F6@172.18.193.251
CSeq: 104 NOTIFY
Max-Forwards: 70
Date: Fri, 01 Mar 2002 00:16:24 GMT
User-Agent: Cisco-SIPGateway/IOS-12.x
Event: kpml
Subscription-State: active
Contact: <sip:172.18.193.251:5060>
Content-Length: 0
//-1/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg:
Received:
NOTIFY sip:172.18.193.251:5060 SIP/2.0
Via: SIP/2.0/UDP 172.18.193.250:5060;branch=z9hG4bK6111
From: <sip:8888@172.18.193.250>;tag=39497C-2EA
To: <sip:172.18.193.251>;tag=EA330-F6
Call-ID: 57633F68-2BE011D6-8013D46B-B4F9B5F6@172.18.193.251
CSeq: 102 NOTIFY
Max-Forwards: 70
Date: Fri, 01 Mar 2002 01:02:51 GMT
User-Agent: Cisco-SIPGateway/IOS-12.x
Event: kpml
Subscription-State: active
Contact: <sip:172.18.193.250:5060>
Content-Length: 0
...
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode: encode_data=0x64E25D00
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode_context_create: chunk_size=2k, max_allowed=16k
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode_context_create: context=0x64FADC10, mp=0x64AFBBE0
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_build_response:
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode: malloc xml_buf=0x645E910C, length=112
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_build_response:
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_build_response: length=73, buffp=0x645E917B
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode: rc=0, encoded str=<?xml version="1.0"
encoding="UTF-8"?><kpml-response version="1.0" code="200" text="OK" digits="1" tag="dtmf"/>
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode_context_free:
kpml_encode_context_free:mem_mgr_mempool_free: mem_refcnt(64AFBBE0)=0 - mempool cleanup
//-1/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg:
Sent:
NOTIFY sip:172.18.193.250:5060 SIP/2.0
Via: SIP/2.0/UDP 172.18.193.251:5060;branch=z9hG4bK1117DE
From: <sip:172.18.193.251>;tag=EA330-F6
To: <sip:8888@172.18.193.250>;tag=39497C-2EA
Call-ID: 57633F68-2BE011D6-8013D46B-B4F9B5F6@172.18.193.251
CSeq: 105 NOTIFY
Max-Forwards: 70
Date: Fri, 01 Mar 2002 00:37:33 GMT
User-Agent: Cisco-SIPGateway/IOS-12.x
Event: kpml
Subscription-State: active
Contact: <sip:172.18.193.251:5060>
Content-Type: application/kpml-response+xml
Content-Length: 113
<?xml version="1.0" encoding="UTF-8"?><kpml-response version="1.0" code="200" text="OK"
```

```
digits="1" tag="dtmf"/>
/-1/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg:
Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP 172.18.193.251:5060;branch=z9hG4bK1117DE
From: <sip:172.18.193.251>;tag=EA330-F6
To: <sip:8888@172.18.193.250>;tag=39497C-2EA
Date: Fri, 01 Mar 2002 01:24:08 GMT
Call-ID: 57633F68-2BE011D6-8013D46B-B4F9B5F6@172.18.193.251
CSeq: 105 NOTIFY
Content-Length: 0
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode: encode_data=0x64E25D00
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode_context_create: chunk_size=2k, max_allowed=16k
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode_context_create: context=0x651E8084, mp=0x65501720
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_build_response:
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode: malloc xml_buf=0x645E910C, length=112
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_build_response:
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_build_response: length=73, buffp=0x645E917B
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode: rc=0, encoded str=<?xml version="1.0"
encoding="UTF-8"?><kpml-response version="1.0" code="200" text="OK" digits="2" tag="dtmf"/>
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode_context_free:
kpml_encode_context_free:mem_mgr_mempool_free: mem_refcnt(65501720)=0 - mempool cleanup
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode: encode_data=0x656F9128
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode_context_create: chunk_size=2k, max_allowed=16k
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode_context_create: context=0x651E8084, mp=0x6488B6CC
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_build_response:
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode: malloc xml_buf=0x645E910C, length=112
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_build_response:
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_build_response: length=73, buffp=0x645E917B
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode: rc=0, encoded str=<?xml version="1.0"
encoding="UTF-8"?><kpml-response version="1.0" code="200" text="OK" digits="3" tag="dtmf"/>
//-1/xxxxxxxxxxxx/KPML/Builder/kpml_encode_context_free:
kpml_encode_context_free:mem_mgr_mempool_free: mem_refcnt(6488B6CC)=0 - mempool cleanup
//-1/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg:
Sent:
NOTIFY sip:172.18.193.250:5060 SIP/2.0
Via: SIP/2.0/UDP 172.18.193.251:5060;branch=z9hG4bK12339
From: <sip:172.18.193.251>;tag=EA330-F6
To: <sip:8888@172.18.193.250>;tag=39497C-2EA
Call-ID: 57633F68-2BE011D6-8013D46B-B4F9B5F6@172.18.193.251
CSeq: 106 NOTIFY
Max-Forwards: 70
Date: Fri, 01 Mar 2002 00:37:44 GMT
User-Agent: Cisco-SIPGateway/IOS-12.x
Event: kpml
Subscription-State: active
Contact: <sip:172.18.193.251:5060
Content-Type: application/kpml-response+xml
Content-Length: 113
<?xml version="1.0" encoding="UTF-8"?><kpml-response version="1.0" code="200" text="OK"
digits="2" tag="dtmf"/>
/-1/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg:
Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP 172.18.193.251:5060;branch=z9hG4bK12339
From: <sip:172.18.193.251>;tag=EA330-F6
To: <sip:8888@172.18.193.250>;tag=39497C-2EA
Date: Fri, 01 Mar 2002 01:24:20 GMT
Call-ID: 57633F68-2BE011D6-8013D46B-B4F9B5F6@172.18.193.251
CSeq: 106 NOTIFY
Content-Length: 0
...
```

Related Commands

Command	Description
show sip-ua calls	Verifies that the DTMF method is SIP-KPML.

debug kron

To display debugging messages about Command Scheduler policies or occurrences, use the **debug kron** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug kron {**all**| **exec-cli**| **info**| **major**}

no debug kron {**all**| **exec-cli**| **info**| **major**}

Syntax Description

all	Displays all debugging output about Command Scheduler policy lists or occurrences.
exec-cli	Displays detailed debugging output about Command Scheduler policy list command-line interface (CLI) commands.
info	Displays debugging output about Command Scheduler policy lists, occurrence warnings, or progress information.
major	Displays debugging output about Command Scheduler policy list or occurrence failures.

Command Default

If no keyword is specified, all debugging messages are displayed.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(1)	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SRC	This command was integrated into Cisco IOS Release 12.2(33)SRC.
12.2(33)SB	This command was integrated into Cisco IOS Release 12.2(33)SB.
12.2(33)SXI	This command was integrated into Cisco IOS Release 12.2(33)SXI.

Usage Guidelines

Use the **debug kron** command to display the output of a scheduled EXEC **show** command on the console.

Examples

The following example shows debugging messages for the EXEC CLI **show version** after the CLI was run at a scheduled interval:

```
Router# debug kron exec-cli

Kron cli occurrence messages debugging is on
2w6d: Call parse_cmd 'show version'
2w6d: Kron CLI return 0
'
**CLI 'show version':
Cisco Internetwork Operating System Software IOS (tm) C2600 Software (C2600-I-M
```

Related Commands

Command	Description
show kron schedule	Displays the status and schedule information for Command Scheduler occurrences.

debug l2ctrl

To enable debugging for Layer 2 Control (L2CTRL), use the **debug l2ctrl** command in privileged EXEC mode. To disable debugging for L2CTRL, use the **no** form of this command.

debug l2ctrl {**all**| **evc**| **pm**| **registry**}

no debug l2ctrl {**all**| **evc**| **pm**| **registry**}

Syntax Description

all	Displays all L2CTRL debugging messages.
evc	Displays Ethernet virtual circuit (EVC) and L2CTRL messages.
pm	Displays switch PM and L2CTRL messages.
registry	Displays L2CTRL registries.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SRD	This command was introduced.

Examples

The following example shows how to enable debugging of all L2CTRL related events:

```
Router# debug l2ctrl all
```

Related Commands

Command	Description
debug ethernet l2ctrl	Enables Ethernet L2CTRL debugging messages.

debug l2fib

To enable the logging of Layer 2 Forwarding Information Base (L2FIB) debug messages, use the **debug l2fib** command in privileged EXEC mode. To disable the debugging, use the **no** form of this command.

debug l2fib {**addr** [**unicast**| **multicast**]| **all**| **bridge-domain [port]**| **event**| **error**| **ha**| **l2port**| **mlrib**| **olist**| **otv tunnel** {**decap**| **encap**}}

no debug l2fib {**addr** [**unicast**| **multicast**]| **all**| **bridge-domain [port]**| **event**| **error**| **ha**| **l2port**| **mlrib**| **olist**| **otv tunnel** {**decap**| **encap**}}

Syntax Description

addr	Enables logging of unicast or multicast object-specific debug messages.
unicast	(Optional) Enables logging of unicast object-specific debug messages.
multicast	(Optional) Enables logging of multicast object-specific debug messages.
all	Enables logging of all L2FIB debug messages.
bridge-domain	Enables logging of bridge-domain object-specific debug messages.
port	Enables logging of bridge-domain port object-specific debug messages.
event	Enables logging of event debug messages.
error	Enables logging of the error debug messages.
ha	Enables logging of high availability (HA) events.
l2port	Enables logging of Layer 2 port object-specific debug messages.
mlrib	Enables logging of Multilayer Routing Information Base (MLRIB) interactions.
olist	Enables logging of output list object-specific debug messages.
otv tunnel	Enables logging of Overlay Transport Virtualization (OTV) tunnel object-specific debug messages.
decap	Enables logging of OTV tunnel decap object-specific debug messages.
encap	Enables logging of OTV tunnel encap object-specific debug messages.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.5S	This command was introduced.

Examples

The following is sample output from the **debug l2fib all** command:

```
Router# debug l2fib all

[11/11/11 19:57:17.256 169DFD 276] L2FIB-MCAST-DEBUG: l2fib_mcast_obj_handle_nh_list:
Received next hop 70.1.1.2, Intf Ov47, owner 0x400, opcode 0x0, flags 0x2, for source
120.0.7.51, group 225.1.8.51, bd 1864
[11/11/11 19:57:17.256 169DFE 276] L2FIB-MCAST-DEBUG: l2fib_mcast_obj_add_oif: Added OIF
ED 70.1.1.2, Intf Ov47, to src 120.0.7.51, grp 225.1.8.51 BD 1864.
[11/11/11 19:57:17.256 169DFF 276] L2FIB-MCAST-DEBUG: l2fib_mcast_obj_add_oif: Found existing
 Otv DG mapping 232.1.47.2, map count 2 for source 120.0.7.51, group 225.1.8.51, bridge
domain 1864, OIF 70.1.1.2, Intf Ov47.
[11/11/11 19:57:17.256 169E00 276] L2FIB-HA-DEBUG: l2fib_ha_encode_mcast_nh_tlv: Encode Otv
 ED receiver port 70.1.1.2, Intf Ov47
[11/11/11 19:57:17.256 169E01 276] L2FIB-HA-DEBUG: l2fib_mcast_obj_chkp: Sync multicast
with src 120.0.7.51, grp 225.1.8.51 BD 1864, rcvrs 1, Otv DG map 1, oper 0x4.
[11/11/11 19:57:17.256 169E02 276] L2FIB-HA-DEBUG: l2fib_ha_enqueue_message: Enqueued message
 to hold queue 0
[11/11/11 19:57:17.256 169E03 276] L2FIB-MCAST-DEBUG: l2fib_mcast_obj_handle_s_g: Received
 source 120.0.7.53, group 225.1.8.53, bridge domain 1866, next hop count 1.
[11/11/11 19:57:17.256 169E04 276] L2FIB-MCAST-DEBUG: l2fib_mcast_obj_handle_nh_list:
Received next hop 70.1.1.2, Intf Ov47, owner 0x400, opcode 0x0, flags 0x2, for source
120.0.7.53, group 225.1.8.53, bd 1866
[11/11/11 19:57:17.256 169E05 276] L2FIB-MCAST-DEBUG: l2fib_mcast_obj_add_oif: Added OIF
ED 70.1.1.2, Intf Ov47, to src 120.0.7.53, grp 225.1.8.53 BD 1866.
[11/11/11 19:57:17.256 169E06 276] L2FIB-MCAST-DEBUG: l2fib_mcast_obj_add_oif: Found existing
 Otv DG mapping 232.1.47.4, map count 2 for source 120.0.7.53, group 225.1.8.53, bridge
domain 1866, OIF 70.1.1.2, Intf Ov47.
[11/11/11 19:57:17.256 169E07 276] L2FIB-HA-DEBUG: l2fib_ha_encode_mcast_nh_tlv: Encode Otv
 ED receiver port 70.1.1.2, Intf Ov47
[11/11/11 19:57:17.256 169E08 276] L2FIB-HA-DEBUG: l2fib_mcast_obj_chkp: Sync multicast
with src 120.0.7.53, grp 225.1.8.53 BD 1866, rcvrs 1, Otv DG map 1, oper 0x4.
[11/11/11 19:57:17.256 169E09 276] L2FIB-HA-DEBUG: l2fib_ha_enqueue_message: Enqueued message
 to hold queue 0
[11/11/11 19:57:17.256 169E0A 276] L2FIB-MCAST-DEBUG: l2fib_mcast_obj_handle_s_g: Received
 source 120.0.7.58, group 225.1.8.58, bridge domain 1871, next hop count 1.
```

Related Commands

Command	Description
show l2fib	Displays information about L2FIB.

debug l2relay events

To start debugging of Layer 2 Relay events, use the **debug l2relay events** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command (SGSN D-node only).

debug l2relay events

no debug l2relay events

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(1)GA	This command was introduced.
12.1(3)T	This command was integrated into Cisco IOS Release 12.1(3)T.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

The SGSN module uses the proprietary Layer 2 Relay protocol in conjunction with the intra-Serving GPRS Support Node (iSGSN) protocol for communication between the SGSN-datacom (SGSN-D) and SGSN-telecom (SGSN-T) units that comprise the SGSN.

For debugging purposes, it might also be useful to trace Layer 2 Relay packets. To display information about Layer 2 Relay packets, use the **debug l2relay packets** command.

Normally you will not need to use the **debug l2relay events** or **debug l2relay packets** commands. If problems with the SGSN are encountered, Cisco technical support personnel may request that issue the command.

Caution Because the **debug l2relay events**command generates a substantial amount of output, use it only when traffic on the GPRS network is low, so other activity on the system is not adversely affected.

Examples

The following example enables the display of Layer 2 Relay events:

```
Router# debug l2relay events
```

Related Commands

Command	Description
debug 12relay packets	Displays Layer 2 Relay packets (SGSN D-node only).

debug l2relay packets

To display information about Layer 2 Relay packets, use the **debug l2relay packets** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command (SGSN D-node only).

debug l2relay packets

no debug l2relay packets

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(1)GA	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.1(3)T	This command was integrated into Cisco IOS Release 12.1(3)T.

Usage Guidelines

Use the **debug l2relay packets** command to display information about Layer 2 Relay packets.

The SGSN module uses the proprietary Layer 2 Relay protocol in conjunction with the intra-Serving GPRS Support Node (iSGSN) protocol for communication between the SGSN-datacom (SGSN-D) and SGSN-telecom (SGSN-T) units that comprise the SGSN.

For debugging purposes, it might also be useful to trace Layer 2 Relay events. To display information about Layer 2 Relay events, use the **debug l2relay events** command.

Normally you will not need to use the **debug l2relay packets**or **debug l2relay events** command. If problems with the SGSN are encountered, Cisco technical support personnel may request that you issue the command.

Caution Because the **debug l2relay packets**command generates a significant amount of output, use it only when traffic on the GPRS network is low, so other activity on the system is not adversely affected.

Examples

The following example enables the display of Layer 2 Relay packets:

```
Router# debug l2relay packets
```

Related Commands

Command	Description
debug ip igmp	Displays Layer 2 Relay events (SGSN D-node only).

debug l2tp

To enable debugging of Layer 2 Tunneling Protocol (L2TP) information, use the **debug l2tp**command in privileged EXEC mode. To disable L2TP debugging, use the **no** form of this command.

debug l2tp {**all**| **application**| **brief**| **db** {**error**| **event**| **lookup**}| **error**| **event**| **export**| **l2tun**| **packet** {**brief**| **detail**| **error**| **event**}| **route**| **seq [brief]**| **snmp**| **timer**}

no debug l2tp {**all**| **application**| **brief**| **db** {**error**| **event**| **lookup**}| **error**| **event**| **export**| **l2tun**| **packet** {**brief**| **detail**| **error**| **event**}| **route**| **seq [brief]**| **snmp**| **timer**}

Syntax Description

all	Enables the most commonly used L2TP debugs.
application	Enables L2TP application information debugs.
brief	Enables L2TP debug information in a single line.
db	Enables L2TP database debugs.
error	Enables L2TP error debugs.
event	Enables L2TP event debugs.
lookup	Enables L2TP database lookup.
export	Enables L2TP external data and command-line interface (CLI) debugs.
l2tun	Enables Layer 2 tunnel (L2Tun) socket application programming interface (API) debugs.
packet	Enables L2TP packet information debugs.
detail	Enables L2TP packet dump details debugs.
route	Enables L2TP route watch debugs.
seq	Enables extra sequencing debugs.
brief	(Optional) Enables L2TP one-line sequencing debugs.
snmp	Enables L2TP Simple Network Management Protocol (SNMP) event debugs.
timer	Enables L2TP timer debugs.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(2)T	This command was introduced.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB.
12.2(33)SB	This command was integrated into Cisco IOS Release 12.2(33)SB.
15.0(1)M	This command was modified. The **application** and **brief** keywords were added.
15.0(1)S	This command was modified. The **snmp** and **route** keywords were added.

Usage Guidelines

Use the **debug l2tp** command to troubleshoot L2TP operations.

Examples

The following example shows how to enable L2TP debugging:

```
Router> enable
Router# debug l2tp all
L2TP most commonly used debugs debugging is on
Router# debug l2tp application
L2TP application debugs debugging is on
Router# debug l2tp brief
L2TP brief, one line debugs debugging is on
Router# debug l2tp db lookup

L2TP database lookups debugging is on
Router# debug l2tp error
L2TP errors debugging is on
Router# debug l2tp seq
L2TP sequencing debugging is on
Router# debug l2tp snmp
L2TP SNMP events debugging is on
```

The following sample output of the **show debugging** command displays the debugs enabled for L2TP. The field descriptions are self-explanatory.

```
Router# show debugging

L2TP:
  L2TP packet events debugging is on
  L2TP packet errors debugging is on
  L2TP packet detail debugging is on
  L2TP errors debugging is on
  L2TP events debugging is on
  L2TP L2TUN socket API debugging is on
  L2TP sequencing debugging is on
  L2TP export data to applications and cli debugging is on
  L2TP route watch debugging is on
  L2TP timers debugging is on
  L2TP brief, one line debugs debugging is on
  L2TP application debugs debugging is on
```

```
L2TP database lookups debugging is on
L2TP SNMP events debugging is on
```

Related Commands

Command	Description
show debugging	Displays information about the types of debugging that are enabled for your router.

debug l2tp redundancy

To enable the display of information on Layer 2 Tunneling Protocol (L2TP) sessions that contain redundancy status, use the **debug l2tp redundancy**command in user or privileged EXEC mode. To disable this debugging, use the **no** form of this command.

debug l2tp redundancy {**cf**| **detail**| **error**| **event**| **fsm**| **resync**| **rf**}

no debug l2tp redundancy

Syntax Description

cf	Displays L2TP redundancy-facility (cf) events.
detail	Displays L2TP redundancy details.
error	Displays L2TP redundancy errors.
event	Displays L2TP redundancy events.
fsm	Displays L2TP redundancy forwarding-service manager (fsm) events.
resync	Displays L2TP redundancy resynchronizations.
rf	Displays L2TP redundancy-facility (rf) events.

Command Modes

User EXEC (>) Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 2.2.	This command was introduced in Cisco IOS XE Release 2.2.

Usage Guidelines

Use the **debug l2tp redundancy**command in privileged EXEC mode to display a list of redundancy events and errors.

Use the **show l2tp redundancy**command in privileged EXEC mode to display information on the state of the Layer 2 Tunneling Protocol (L2TP) or a specific L2TP session redundancy data.

Examples

The following example shows how to display a debug of redundancy events during the setup and termination of an L2TP High Availability (HA) tunnel for a L2TP Network Server (LNS) active Route Processor (RP):

```
LNS1> debug
 enable
LNS1# debug
```

```
 l2tp
 redundancy
 cf
L2TP redundancy cf debugging is on
LNS1# debug
 l2tp
 redundancy
 detail
L2TP redundancy details debugging is on
LNS1# debug
 l2tp
 redundancy
 error
L2TP redundancy errors debugging is on
LNS1# debug
 l2tp
 redundancy
 event
L2TP redundancy events debugging is on
LNS1# debug
 l2tp
 redundancy
 fsm
L2TP redundancy fsm debugging is on
LNS1# debug
 l2tp
 redundancy
 resync
L2TP redundancy resync debugging is on
LNS1# debug
 l2tp
 redundancy
 rf
L2TP redundancy rf debugging is on
LNS1#
*Aug 26 18:00:00.467: %SYS-5-CONFIG_I: Configured from console by console
LNS1#
*Aug 26 18:00:45.631: L2TP tnl   01000:________: CCM initialized CCM session
*Aug 26 18:00:45.631: : L2TP HA:CC playback chkpt skipped, CC not doing HA
*Aug 26 18:00:45.711: : L2TP HA FSM:Receive proto FSM event 19
*Aug 26 18:00:45.711: : L2TP HA FSM:Receive RxSCCRQ
*Aug 26 18:00:45.711: : L2TP HA:lcm_cc alloc: l2tp_cc 070B45B8, lcm_cc 02FE55E8
*Aug 26 18:00:45.711: : L2TP HA FSM:FSM-CC ev Rx-SCCRQ
*Aug 26 18:00:45.711: : L2TP HA FSM:FSM-CC    Idle->Wt-ChkptSidRmt
*Aug 26 18:00:45.711: : L2TP HA FSM:FSM-CC do Block-Tx-AckSCCRQ
*Aug 26 18:00:45.711: : L2TP HA FSM:Checkpoint Two Cc IDs
*Aug 26 18:00:45.711: L2TP HA CF: Chkpt send: s/c id 0/52631, BothCcId, seq 0, ns/nr 0/0,
rid 51583, len 52; flush = 1, ctr 1
*Aug 26 18:00:45.711: 01000:0000CD97: L2TP HA:Enqueue peer Ns 0 to ns_q, seq 1 (q sz 0)
*Aug 26 18:00:45.711: L2TP tnl   01000:0000CD97: Encoding SCCRQ-IN CHKPT
*Aug 26 18:00:45.711: L2TP tnl   01000:0000CD97: Tx CHKPT
*Aug 26 18:00:45.739: L2TP tnl   01000:0000CD97: Encoding SCCRP-OUT CHKPT
*Aug 26 18:00:45.739: L2TP tnl   01000:0000CD97: Tx CHKPT
*Aug 26 18:00:45.739: : L2TP HA:Adjust local window size to 10
*Aug 26 18:00:45.739: 01000:0000CD97: L2TP HA FSM:Receive proto TxCM event SCCRP
*Aug 26 18:00:45.739: : L2TP HA FSM:Receive TxSCCRP
LNS1#
*Aug 26 18:00:45.739: : L2TP HA FSM:FSM-CC ev Tx-SCCRP
*Aug 26 18:00:45.739: : L2TP HA FSM:FSM-CC    Wt-ChkptSidRmt->WtCcIdRmt2
*Aug 26 18:00:45.739: : L2TP HA FSM:FSM-CC do Block-Tx-SCCRP
*Aug 26 18:00:45.739: 01000:0000CD97: L2TP HA FSM:Found blocked RxSCCRQ, seq_num 1
*Aug 26 18:00:45.739: 01000:0000CD97: L2TP HA FSM:Queued SCCRP to CC hold_q
*Aug 26 18:00:46.863: : L2TP HA FSM:CHKPT status callback: status 0, len 56
*Aug 26 18:00:46.863: : L2TP HA FSM:Context s/c id 0/52631, BothCcId, seq 1, ns/nr 0/0, rid
 51583, len 52
*Aug 26 18:00:46.863: L2TP HA CF: Rcvd status s/c id 0/52631, BothCcId, seq 1, ns/nr 0/0,
rid 51583, len 52
*Aug 26 18:00:46.863: L2TP HA CF: Rcvd status 0: len 56
*Aug 26 18:00:46.863: L2TP HA CF: Status content s/c id 0/52631, BothCcId, seq 1, ns/nr
0/0, rid 51583, len 52
*Aug 26 18:00:46.863: 01000:0000CD97: L2TP HA FSM:Recv chkpt ack: s/c id 0/52631, BothCcId,
 seq 1, ns/nr 0/0, rid 51583, len 52
```

```
*Aug 26 18:00:46.863: : L2TP HA FSM:Receive CC-ChkptAck
*Aug 26 18:00:46.863: : L2TP HA FSM:FSM-CC ev Rx-CkpACK-CcID-Rmt
*Aug 26 18:00:46.863: : L2TP HA FSM:FSM-CC    WtCcIdRmt2->Wt-RxSccn
*Aug 26 18:00:46.863: : L2TP HA FSM:FSM-CC do Allow-Tx-SCCRP2
*Aug 26 18:00:46.863: : L2TP HA FSM:Received Chkpt of local + remote CC ID
*Aug 26 18:00:46.863: 01000:0000CD97: L2TP HA:Try to remove from CC's ns_q: seq num 1
(current Ns 1)
*Aug 26 18:00:46.863: 01000:0000CD97: L2TP HA:Ns entry to remove: found (current Ns 1)
*Aug 26 18:00:46.863: 01000:0000CD97: L2TP HA:Advance peer Nr to 1 (ns_q sz 0)
*Aug 26 18:00:46.863: 01000:0000CD97: L2TP HA:CC send all unblocked if can
LNS1#
*Aug 26 18:00:46.863: 01000:0000CD97: L2TP HA:CC send one blocked CM (SCCRP): ns 0 (0), nr
 1
*Aug 26 18:00:46.863: L2TP HA CF: O SCCRP 51583/0 ns/nr 0/1
*Aug 26 18:00:47.867: 01000:0000CD97: L2TP HA FSM:Receive Cm-Ack
*Aug 26 18:00:47.867: 01000:0000CD97: L2TP HA FSM:Receive CC Cm-Ack
*Aug 26 18:00:47.867: : L2TP HA FSM:FSM-CC ev Rx-CmACK
*Aug 26 18:00:47.867: : L2TP HA FSM:FSM-CC    in Wt-RxSccn
*Aug 26 18:00:47.867: : L2TP HA FSM:FSM-CC do Ignore
*Aug 26 18:00:47.867: 01000:0000CD97: L2TP HA FSM:Ignore event
*Aug 26 18:00:47.867: 01000:0000CD97: L2TP HA FSM:Check for Ns/Nr update 1, peer 1
*Aug 26 18:00:47.867: 01000:0000CD97: L2TP HA FSM:Receive peer Ns/Nr update (1,0/1,1, int
1, rx 1, 1) (ns_q sz 0)
*Aug 26 18:00:47.867: 01000:0000CD97: L2TP HA FSM:Peer Ns 1 (1), Nr 1 (ns_q sz 0)
*Aug 26 18:00:48.087: 01000:0000CD97: L2TP HA FSM:Check for Ns/Nr update 1, peer 1
*Aug 26 18:00:48.087: 01000:0000CD97: L2TP HA FSM:Receive peer Ns/Nr update (1,0/1,1, int
1, rx 1, 1) (ns_q sz 0)
*Aug 26 18:00:48.087: 01000:0000CD97: L2TP HA FSM:Peer Ns 1 (2), Nr 1 (ns_q sz 0)
*Aug 26 18:00:48.087: : L2TP HA FSM:Receive proto FSM event 21
*Aug 26 18:00:48.087: 01000:0000CD97: L2TP HA FSM:Receive RxSCCCN
*Aug 26 18:00:48.087: : L2TP HA FSM:FSM-CC ev Rx-SCCCN
*Aug 26 18:00:48.087: : L2TP HA FSM:FSM-CC    Wt-RxSccn->WtCcsUp
*Aug 26 18:00:48.087: : L2TP HA FSM:FSM-CC do Allow-Tx-AckSCCCN
*Aug 26 18:00:48.087: 01000:0000CD97: L2TP HA FSM:Allow TxSCCCN-ACK
*Aug 26 18:00:48.087: 01000:0000CD97: L2TP HA FSM:Receive CcUp
*Aug 26 18:00:48.087: : L2TP HA FSM:FSM-CC ev Proto CcUp
*Aug 26 18:00:48.087: : L2TP HA FSM:FSM-CC    WtCcsUp->Wt-CkptCcUp
*Aug 26 18:00:48.087: : L2TP HA FSM:FSM-CC do Chkpt-CcUp2
*Aug 26 18:00:48.087: : L2TP HA FSM:Checkpoint CcUp
*Aug 26 18:00:48.087: L2TP HA CF: Chkpt send: s/c id 0/52631, CcUp, seq 0, ns/nr 1/1, rid
0, len 52; flush = 1, ctr 2
*Aug 26 18:00:48.091: L2TP tnl   01000:0000CD97: CCM added sync data
*Aug 26 18:00:48.095: 01000:0000CD97: L2TP HA FSM:Check for Ns/Nr update 2, peer 1
*Aug 26 18:00:48.095: 01000:0000CD97: L2TP HA FSM:Receive peer Ns/Nr update (2,1/1,1, int
2, rx 1, 2) (ns_q sz 0)
*Aug 26 18:00:48.095: 01000:0000CD97: L2TP HA FSM:Peer Ns 2 (3), Nr 1 (ns_q sz 0)
*Aug 26 18:00:48.095: L2TP _____:01000:000036F8: Encoding ICRQ-IN CHKPT
*Aug 26 18:00:48.095: L2TP _____:01000:000036F8: Tx CHKPT
*Aug 26 18:00:48.095: : L2TP HA FSM:Receive proto FSM event 3
*Aug 26 18:00:48.095: : L2TP HA FSM:Receive RxICRQ
*Aug 26 18:00:48.095: _____:01000:000036F8: L2TP HA FSM:  Using ICRQ FSM
*Aug 26 18:00:48.095: _____:01000:000036F8: L2TP HA FSM:FSM-Sn ev created
*Aug 26 18:00:48.095: _____:01000:000036F8: L2TP HA FSM:FSM-Sn    Init->Idle
*Aug 26 18:00:48.095: _____:01000:000036F8: L2TP HA FSM:FSM-Sn do none
*Aug 26 18:00:48.095: _____:01000:000036F8: L2TP HA FSM:FSM-Sn ev Rx-xCRQ
*Aug 26 18:00:48.095: _____:01000:000036F8: L2TP HA FSM:FSM-Sn    Idle->Wt-ChkptSidRmt
*Aug 26 18:00:48.095: _____:01000:000036F8: L2TP HA FSM:FSM-Sn do Block-Tx-AckXCRQ
*Aug 26 18:00:48.095: _____:01000:000036F8: L2TP HA FSM:Checkpoint TwoSessionIDs
*Aug 26 18:00:48.095: L2TP HA CF: Chkpt send: s/c id 14072/52631, BothSesId, seq 0, ns/nr
1/2, rid 40276, len 52; flush = 1, ctr 3
*Aug 26 18:00:48.095: _____:01000:000036F8: L2TP HA:Enqueue peer Ns 2 to ns_q, seq 3 (q sz
 0)
*Aug 26 18:00:48.131: : L2TP HA:Try to buffer sock msg type 19
*Aug 26 18:00:48.131: : L2TP HA:Buffering skipped
*Aug 26 18:00:48.131: L2TP _____:01000:000036F8: Encoding ICRP-OUT CHKPT
*Aug 26 18:00:48.131: L2TP _____:01000:000036F8: Tx CHKPT
*Aug 26 18:00:48.131: 01000:0000CD97: L2TP HA FSM:Receive proto TxCM event ICRP
*Aug 26 18:00:48.131: _____:_____:000036F8: L2TP HA FSM:Receive TxICRP
*Aug 26 18:00:48.131: _____:01000:000036F8: L2TP HA FSM:FSM-Sn ev Tx-xCRP
*Aug 26 18:00:48.131: _____:01000:000036F8: L2TP HA FSM:FSM-Sn    Wt-ChkptSidRmt->Wt-SesIdRmt2
*Aug 26 18:00:48.131: _____:01000:000036F8: L2TP HA FSM:FSM-Sn do Block-Tx-xCRP
*Aug 26 18:00:48.131: _____:01000:000036F8: L2TP HA FSM:Found blocked RxICRQ, seq_num 3
LNS1#
```

```
*Aug 26 18:00:48.131: _____:01000:000036F8: L2TP HA FSM:Queued xCRP to session hold_q
*Aug 26 18:00:48.131: : L2TP HA:Try to buffer sock msg type 23
*Aug 26 18:00:48.131: : L2TP HA:CC not in resync state, buffering skipped
*Aug 26 18:00:49.115: 01000:0000CD97: L2TP HA FSM:Check for Ns/Nr update 2, peer 1
*Aug 26 18:00:49.115: 01000:0000CD97: L2TP HA FSM:Receive peer Ns/Nr update (2,2/1,1, int
3, rx 1, 3) (ns_q sz 1)
*Aug 26 18:00:49.211: : L2TP HA FSM:CHKPT status callback: status 0, len 56
*Aug 26 18:00:49.211: : L2TP HA FSM:Context s/c id 0/52631, CcUp, seq 2, ns/nr 1/1, rid 0,
 len 52
*Aug 26 18:00:49.211: : L2TP HA FSM:CHKPT status callback: status 0, len 56
*Aug 26 18:00:49.211: : L2TP HA FSM:Context s/c id 14072/52631, BothSesId, seq 3, ns/nr
1/2, rid 40276, len 52
*Aug 26 18:00:49.211: L2TP HA CF: Rcvd status s/c id 0/52631, CcUp, seq 2, ns/nr 1/1, rid
0, len 52
*Aug 26 18:00:49.211: L2TP HA CF: Rcvd status 0: len 56
*Aug 26 18:00:49.211: L2TP HA CF: Status content s/c id 0/52631, CcUp, seq 2, ns/nr 1/1,
rid 0, len 52
*Aug 26 18:00:49.211: 01000:0000CD97: L2TP HA FSM:Recv chkpt ack: s/c id 0/52631, CcUp, seq
 2, ns/nr 1/1, rid 0, len 52
*Aug 26 18:00:49.211: : L2TP HA FSM:Receive CC-ChkptAck
*Aug 26 18:00:49.211: : L2TP HA FSM:FSM-CC ev Rx-CkpACK-CcUp
*Aug 26 18:00:49.211: : L2TP HA FSM:FSM-CC    Wt-CkptCcUp->ProcCcsUp
*Aug 26 18:00:49.211: : L2TP HA FSM:FSM-CC do Proc-ChpACK-CcUp2
*Aug 26 18:00:49.211: : L2TP HA FSM:Received chkpt ACK of CcUp
*Aug 26 18:00:49.211: L2TP HA CF: Rcvd status s/c id 14072/52631, BothSesId, seq 3, ns/nr
1/2, rid 40276, len 52
*Aug 26 18:00:49.211: L2TP HA CF: Rcvd status 0: len 56
*Aug 26 18:00:49.211: L2TP HA CF: Status content s/c id 14072/52631, BothSesId, seq 3, ns/nr
 1/2, rid 40276, len 52
*Aug 26 18:00:49.211: 01000:0000CD97: L2TP HA FSM:Recv chkpt ack: s/c id 14072/52631,
BothSesId, seq 3, ns/nr 1/2, rid 40276, len 52
*Aug 26 18:00:49.211: _____:_____:000036F8: L2TP HA FSM:Receive Session-ChkptAck
*Aug 26 18:00:49.211: _____:01000:000036F8: L2TP HA FSM:FSM-Sn ev Rx-CktACK-SesID-Rmt
*Aug 26 18:00:49.211: _____:01000:000036F8: L2TP HA FSM:FSM-Sn    Wt-SesIdRmt2->Wt-RxXccn
*Aug 26 18:00:49.211: _____:01000:000036F8: L2TP HA FSM:FSM-Sn do Allow-Tx-xCRP
*Aug 26 18:00:49.211: 01000:0000CD97: L2TP HA:Try to remove from CC's ns_q: seq num 3
(current Ns 3)
*Aug 26 18:00:49.211: _____:01000:000036F8: L2TP HA:Ns entry to remove: found (current Ns
3)
*Aug 26 18:00:49.211: _____:01000:000036F8: L2TP HA:Advance peer Nr to 3 (ns_q sz 0)
*Aug 26 18:00:49.211: _____:01000:000036F8: L2TP HA:Session send all unblocked
*Aug 26 18:00:49.211: 01000:0000CD97: L2TP HA:CC send if can (ICRP): ns 1 (1, 1), nr 3 (3)
*Aug 26 18:00:49.211: L2TP HA CF: O ICRP 51583/40276 ns/nr 1/3
*Aug 26 18:00:49.231: 01000:0000CD97: L2TP HA FSM:Receive Cm-Ack
*Aug 26 18:00:49.231: _____:_____:000036F8: L2TP HA FSM:Receive session Cm-Ack
LNS1#
*Aug 26 18:00:49.231: _____:01000:000036F8: L2TP HA FSM:FSM-Sn ev Rx-CmACK
*Aug 26 18:00:49.231: _____:01000:000036F8: L2TP HA FSM:FSM-Sn    in Wt-RxXccn
*Aug 26 18:00:49.231: _____:01000:000036F8: L2TP HA FSM:FSM-Sn do Ignore
*Aug 26 18:00:49.231: _____:01000:000036F8: L2TP HA FSM:Ignore event
*Aug 26 18:00:49.231: 01000:0000CD97: L2TP HA FSM:Check for Ns/Nr update 3, peer 2
*Aug 26 18:00:49.231: 01000:0000CD97: L2TP HA FSM:Receive peer Ns/Nr update (3,2/2,2, int
3, rx 2, 3) (ns_q sz 0)
*Aug 26 18:00:49.231: 01000:0000CD97: L2TP HA FSM:Peer Ns 3 (3), Nr 2 (ns_q sz 0)
LNS1#
*Aug 26 18:00:50.407: 01000:0000CD97: L2TP HA FSM:Check for Ns/Nr update 3, peer 2
*Aug 26 18:00:50.407: 01000:0000CD97: L2TP HA FSM:Receive peer Ns/Nr update (3,2/2,2, int
3, rx 2, 3) (ns_q sz 0)
*Aug 26 18:00:50.407: 01000:0000CD97: L2TP HA FSM:Peer Ns 3 (4), Nr 2 (ns_q sz 0)
*Aug 26 18:00:50.407: : L2TP HA FSM:Receive proto FSM event 5
*Aug 26 18:00:50.407: _____:_____:000036F8: L2TP HA FSM:Receive RxICCN
*Aug 26 18:00:50.407: _____:01000:000036F8: L2TP HA FSM:FSM-Sn ev Rx-xCCN
*Aug 26 18:00:50.407: _____:01000:000036F8: L2TP HA FSM:FSM-Sn    Wt-RxXccn->Wt-SessUp
*Aug 26 18:00:50.407: _____:01000:000036F8: L2TP HA FSM:FSM-Sn do Allow-Tx-AckXCCN
*Aug 26 18:00:50.407: _____:01000:000036F8: L2TP HA FSM:Allow TxICCN-ACK
*Aug 26 18:00:50.407: L2TP _____:01000:000036F8: Encoding ICCN-IN CHKPT
*Aug 26 18:00:50.407: L2TP _____:01000:000036F8: Tx CHKPT
*Aug 26 18:00:50.407: _____:_____:000036F8: L2TP HA FSM:Receive SessionUp
*Aug 26 18:00:50.407: _____:01000:000036F8: L2TP HA FSM:FSM-Sn ev Proto SessUp
*Aug 26 18:00:50.407: _____:01000:000036F8: L2TP HA FSM:FSM-Sn    Wt-SessUp->Wt-CkptSesUp
*Aug 26 18:00:50.407: _____:01000:000036F8: L2TP HA FSM:FSM-Sn do Chkpt-SesUp2
*Aug 26 18:00:50.407: _____:01000:000036F8: L2TP HA FSM:Checkpoint SessionUP
*Aug 26 18:00:50.407: L2TP HA CF: Chkpt send: s/c id 14072/52631, SesUp, seq 0, ns/nr 2/3,
```

```
 rid 0, len 52; flush = 1, ctr 4
*Aug 26 18:00:51.055: : L2TP HA FSM:CHKPT status callback: status 0, len 56
*Aug 26 18:00:51.055: : L2TP HA FSM:Context s/c id 14072/52631, SesUp, seq 4, ns/nr 2/3,
rid 0, len 52
*Aug 26 18:00:51.055: L2TP HA CF: Rcvd status s/c id 14072/52631, SesUp, seq 4, ns/nr 2/3,
 rid 0, len 52
*Aug 26 18:00:51.055: L2TP HA CF: Rcvd status 0: len 56
*Aug 26 18:00:51.055: L2TP HA CF: Status content s/c id 14072/52631, SesUp, seq 4, ns/nr
2/3, rid 0, len 52
*Aug 26 18:00:51.055: 01000:0000CD97: L2TP HA FSM:Recv chkpt ack: s/c id 14072/52631, SesUp,
 seq 4, ns/nr 2/3, rid 0, len 52
*Aug 26 18:00:51.055: _____:_____:000036F8: L2TP HA FSM:Receive Session-ChkptAck
*Aug 26 18:00:51.055: _____:01000:000036F8: L2TP HA FSM:FSM-Sn ev Rx-CktACK-SesUp
*Aug 26 18:00:51.055: _____:01000:000036F8: L2TP HA FSM:FSM-Sn    Wt-CkptSesUp->Proc-SessUp
*Aug 26 18:00:51.055: _____:01000:000036F8: L2TP HA FSM:FSM-Sn do Proc-ChpACK-SesUp
*Aug 26 18:00:51.055: _____:01000:000036F8: L2TP HA FSM:Received chkpt ACK of SessionUP
*Aug 26 18:00:51.347: %LINK-3-UPDOWN: Interface Virtual-Access2, changed state to up
LNS1#
*Aug 26 18:00:51.635: : L2TP HA:Try to buffer sock msg type 26
*Aug 26 18:00:51.635: : L2TP HA:CC not in resync state, buffering skipped
*Aug 26 18:00:51.659: : L2TP HA:Try to buffer sock msg type 26
*Aug 26 18:00:51.659: : L2TP HA:CC not in resync state, buffering skipped
LNS1#
*Aug 26 18:00:52.363: %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access2,
changed state to up
LNS1#
LNS1# clear
 l2tp
 all
Proceed with clearing all tunnels? [confirm]
LNS1#
*Aug 26 18:01:21.271: 00001:_____:000036F8: L2TP HA FSM:Receive Session-CC-Rm
*Aug 26 18:01:21.271: 00001:_____:000036F8: L2TP HA FSM:Receive SessionRm
*Aug 26 18:01:21.271: 01000:0000CD97: L2TP HA FSM:Receive proto TxCM event StopCCN
*Aug 26 18:01:21.271: 01000:0000CD97: L2TP HA FSM:Receive TxSTOPCCN
*Aug 26 18:01:21.271: : L2TP HA FSM:FSM-CC ev Tx-STOPCCN
*Aug 26 18:01:21.271: : L2TP HA FSM:FSM-CC    ProcCcsUp->Wt-CkptCcDn
*Aug 26 18:01:21.271: : L2TP HA FSM:FSM-CC do Chkpt-CcDwn
*Aug 26 18:01:21.271: 01000:0000CD97: L2TP HA FSM:Receive TxSTOPCCN while CC up
*Aug 26 18:01:21.271: 01000:0000CD97: L2TP HA:CC ns_q cleanup: overall head Ns old/new =
4/4 (Q sz 0)
LNS1#
*Aug 26 18:01:21.271: : L2TP HA FSM:Checkpoint CCDown
*Aug 26 18:01:21.271: L2TP HA CF: Chkpt send: s/c id 0/52631, CcDwn, seq 0, ns/nr 2/3, rid
 0, len 52; flush = 1, ctr 5
*Aug 26 18:01:21.271: 01000:0000CD97: L2TP HA FSM:Queued STOPCCN to cc hold_q
*Aug 26 18:01:21.295: : L2TP HA:Try to buffer sock msg type 22
*Aug 26 18:01:21.295: : L2TP HA:Buffering skipped
*Aug 26 18:01:22.423: : L2TP HA FSM:CHKPT status callback: status 0, len 56
*Aug 26 18:01:22.423: : L2TP HA FSM:Context s/c id 0/52631, CcDwn, seq 5, ns/nr 2/3, rid
0, len 52
*Aug 26 18:01:22.423: L2TP HA CF: Rcvd status s/c id 0/52631, CcDwn, seq 5, ns/nr 2/3, rid
 0, len 52
*Aug 26 18:01:22.423: L2TP HA CF: Rcvd status 0: len 56
*Aug 26 18:01:22.423: L2TP HA CF: Status content s/c id 0/52631, CcDwn, seq 5, ns/nr 2/3,
rid 0, len 52
*Aug 26 18:01:22.423: 01000:0000CD97: L2TP HA FSM:Recv chkpt ack: s/c id 0/52631, CcDwn,
seq 5, ns/nr 2/3, rid 0, len 52
*Aug 26 18:01:22.423: : L2TP HA FSM:Receive CC-ChkptAck
*Aug 26 18:01:22.423: : L2TP HA FSM:FSM-CC ev Rx-CkpACK-CcDwn
*Aug 26 18:01:22.423: : L2TP HA FSM:FSM-CC    Wt-CkptCcDn->Wt-RxStopAck
*Aug 26 18:01:22.423: : L2TP HA FSM:FSM-CC do Allow-Tx-STOPCCN4
*Aug 26 18:01:22.423: : L2TP HA FSM:Received Chkpt of CC removal
*Aug 26 18:01:22.423: 01000:0000CD97: L2TP HA:Try to remove from CC's ns_q: seq num 5
(current Ns 4)
*Aug 26 18:01:22.423: 01000:0000CD97: L2TP HA:Ns entry to remove: not found (current Ns 4)
*Aug 26 18:01:22.423: 01000:0000CD97: L2TP HA:CC send all unblocked if can
*Aug 26 18:01:22.423: 01000:0000CD97: L2TP HA:CC send one blocked CM (SCCRP): ns 2 (2), nr
 4
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA FSM:Receive Cm-Ack
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA FSM:Receive CC Cm-Ack
*Aug 26 18:01:22.451: : L2TP HA FSM:FSM-CC ev Rx-CmACK
*Aug 26 18:01:22.451: : L2TP HA FSM:FSM-CC    Wt-RxStopAck->Wt-CkptCcRm
```

```
*Aug 26 18:01:22.451: : L2TP HA FSM:FSM-CC do ChkptCcRm3
*Aug 26 18:01:22.451: : L2TP HA FSM:Received STOPCCN-ACK while waiting for it, checkpoint
CCRm and remove cc
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA:CC ns_q cleanup: overall head Ns old/new =
4/4 (Q sz 0)
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA FSM:Checkpoint CcRm
*Aug 26 18:01:22.451: L2TP HA CF: Chkpt send: s/c id 0/52631, CcRm, seq 0, ns/nr 3/3, rid
0, len 52; flush = 1, ctr 6
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA FSM:Check for Ns/Nr update 4, peer 3
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA FSM:Receive peer Ns/Nr update (4,3/3,3, int
4, rx 3, 4) (ns_q sz 0)
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA FSM:Peer Ns 4 (4), Nr 3 (ns_q sz 0)
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA FSM:Receive CC-Rm
*Aug 26 18:01:22.451: : L2TP HA FSM:FSM-CC ev Proto CcRm
*Aug 26 18:01:22.451: : L2TP HA FSM:FSM-CC    Wt-CkptCcRm->End
*Aug 26 18:01:22.451: : L2TP HA FSM:FSM-CC do RmCc3
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA FSM:CC destruction after Tx/Rx StopCCN
LNS1#
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA:CC ns_q cleanup: overall head Ns old/new =
4/4 (Q sz 0)
*Aug 26 18:01:22.451: : L2TP HA FSM:Checkpoint CCRm
*Aug 26 18:01:22.451: L2TP HA CF: Chkpt send: s/c id 0/52631, CcRm, seq 0, ns/nr 3/3, rid
0, len 52; flush = 1, ctr 7
*Aug 26 18:01:22.451: : L2TP HA:lcm_cc free: l2tp_cc 070B45B8, lcm_cc 02FE55E8
*Aug 26 18:01:22.451: L2TP tnl     _____:________: CCM setting state to DOWN
*Aug 26 18:01:23.571: : L2TP HA FSM:CHKPT status callback: status 0, len 56
*Aug 26 18:01:23.571: : L2TP HA FSM:Context s/c id 0/52631, CcRm, seq 6, ns/nr 3/3, rid 0,
 len 52
*Aug 26 18:01:23.571: : L2TP HA FSM:CHKPT status callback: status 0, len 56
*Aug 26 18:01:23.571: : L2TP HA FSM:Context s/c id 0/52631, CcRm, seq 7, ns/nr 3/3, rid 0,
 len 52
*Aug 26 18:01:23.571: L2TP HA CF: Rcvd status s/c id 0/52631, CcRm, seq 6, ns/nr 3/3, rid
0, len 52
*Aug 26 18:01:23.571: L2TP HA CF: Rcvd status 0: len 56
*Aug 26 18:01:23.571: L2TP HA CF: Status content s/c id 0/52631, CcRm, seq 6, ns/nr 3/3,
rid 0, len 52
*Aug 26 18:01:23.571: : L2TP HA FSM:Ignore chkpt ACK: CC not found.
LNS1#
*Aug 26 18:01:23.571: L2TP HA CF: Rcvd status s/c id 0/52631, CcRm, seq 7, ns/nr 3/3, rid
0, len 52
*Aug 26 18:01:23.571: L2TP HA CF: Rcvd status 0: len 56
*Aug 26 18:01:23.571: L2TP HA CF: Status content s/c id 0/52631, CcRm, seq 7, ns/nr 3/3,
rid 0, len 52
*Aug 26 18:01:23.571: : L2TP HA FSM:Ignore chkpt ACK: CC not found.
LNS1#
*Aug 26 18:01:35.771: %REDUNDANCY-3-STANDBY_LOST: Standby processor fault
(PEER_DOWN_INTERRUPT)
```

The table below describes the significant fields shown in the **debug l2tp redundancy** command output.

Table 18: debug l2tp redundancy Command Field Descriptions

Field	Description
cf	Number of L2TP checkpointing-facility events (cf-events).
error	Number of L2TP checkpointing errors.
event	Number of L2TP checkpointing events.
fsm	Number of L2TP checkpointing fsm events.
resync	Number of L2TP checkpointing resynchronized events.

Field	Description
rf	Number of L2TP checkpointing redundancy-facility events (rf-events).

Related Commands

Command	Description
debug vpdn redundancy	Displays information about VPDN sessions that have redundancy events and errors.
l2tp sso enable	Enables L2TP HA.
l2tp tunnel resync	Specifies the number of packets sent before waiting for an acknowledgment message.
show l2tp redundancy	Displays L2TP sessions containing redundancy data.
show vpdn redundancy	Displays VPDN sessions containing redundancy data.
sso enable	Enables L2TP HA session for VPDN groups.

debug l2vpn acircuit

To debug errors and events that occur on the Layer 2 VPN (L2VPN) attachment circuits (the circuits between the provider edge [PE] and customer edge [CE] devices), use the **debug l2vpn acircuit** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug l2vpn acircuit {**error**| **event**| **event-trace** *number* [**preserve**]}

no debug l2vpn acircuit {**error**| **event**| **event-trace** *number* [**preserve**]}

Syntax Description

error	Displays errors that occur in attachment circuits.
event	Displays events that occur in attachment circuits.
event-trace	Displays event trace logs.
number	Number of event trace logs to be stored per context.
preserve	Specifies that the event trace logs should not be removed.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.7S	This command was introduced as part of the Multiprotocol Label Switching (MPLS)-based L2VPN command modifications for cross-OS support. This command will replace the **debug acircuit** command in future releases.
15.3(1)S	This command was integrated in Cisco IOS Release 15.3(1)S.

Usage Guidelines

Use the **debug l2vpn acircuit** command to identify provisioning events, setup failures, circuit up and down events, and configuration failures on attachment circuits.

An attachment circuit connects a PE device to a CE device. A device can have many attachment circuits. The attachment circuit manager controls all attachment circuits from one central location. Therefore, when you enable debug messages for an attachment circuit, you receive information about all attachment circuits.

Examples

The following is sample output from the **debug acircuit event** command when an interface is enabled:

```
Device# debug l2vpn acircuit event

*Jan 28 15:19:03.070: ACLIB: ac_cstate() Handling circuit UP for interface Se2/0
```

```
*Jan 28 15:19:03.070: ACLIB [10.0.1.1, 200]: pthru_intf_handle_circuit_up() calling
acmgr_circuit_up
*Jan 28 15:19:03.070: ACLIB [10.0.1.1, 200]: Setting new AC state to Ac-Connecting
*Jan 28 15:19:03.070: ACMGR: Receive <Circuit Up> msg
*Jan 28 15:19:03.070: Se2/0 ACMGR: circuit up event, SIP state chg down to connecting,
action is service request
*Jan 28 15:19:03.070: Se2/0 ACMGR: Sent a sip service request
*Jan 28 15:19:03.070: ACLIB [10.0.1.1, 200]: AC updating switch context.
*Jan 28 15:19:03.070: Se2/0 ACMGR: Rcv SIP msg: resp connect forwarded, hdl 9500001D,
l2ss_hdl 700001E
*Jan 28 15:19:03.070: Se2/0 ACMGR: service connected event, SIP state chg connecting to
connected, action is respond forwarded
*Jan 28 15:19:03.070: ACLIB: pthru_intf_response hdl is 9500001D, response is 1
*Jan 28 15:19:03.070: ACLIB [10.0.1.1, 200]: Setting new AC state to Ac-Connected
```

The following is sample output from the **debug l2vpn acircuit event** command when an interface is disabled:

```
Device# debug l2vpn acircuit event

*Jan 28 15:25:57.014: ACLIB: SW AC interface INTF-DOWN for interface Se2/0
*Jan 28 15:25:57.014: ACLIB [10.0.1.1, 200]: Setting new AC state to Ac-Idle
*Jan 28 15:25:57.014: ACLIB: SW AC interface INTF-DOWN for interface Se2/0
*Jan 28 15:25:57.014: Se2/0 ACMGR: Receive <Circuit Down> msg
*Jan 28 15:25:57.014: Se2/0 ACMGR: circuit down event, SIP state chg connected to end,
action is service disconnect
*Jan 28 15:25:57.014: Se2/0 ACMGR: Sent a sip service disconnect
*Jan 28 15:25:57.014: ACLIB [10.0.1.1, 200]: AC deleting switch context.
*Jan 28 15:25:59.014: %LINK-5-CHANGED: Interface Serial2/0, changed state to
administratively down
*Jan 28 15:25:59.014: ACLIB: ac_cstate() Handling circuit DOWN for interface Se2/0
*Jan 28 15:26:00.014:%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0, changed
state to down
```

The following example shows output from the **debug l2vpn acircuit** command for an xconnect session on a GigabitEthernet interface:

```
Device# debug l2vpn acircuit

23:28:35: ACLIB [10.0.3.201, 5]: SW AC interface UP for GigabitEthernet interface GE2/1/1
23:28:35: ACLIB [10.0.3.201, 5]: pthru_intf_handle_circuit_up() calling acmgr_circuit_up
23:28:35: ACLIB [10.0.3.201, 5]: Setting new AC state to Ac-Connecting
23:28:35: ACLIB [10.0.3.201, 5]: SW AC interface UP for GigabitEthernet interface GE2/1/1
23:28:35: ACLIB [10.0.3.201, 5]: pthru_intf_handle_circuit_up() ignoring up event. Already
 connected or connecting.
23:28:35: ACMGR: Receive <Circuit Up> msg
23:28:35: GE2/1/1 ACMGR: circuit up event, SIP state chg down to connecting, action is
service request
23:28:35: GE2/1/1 ACMGR: Sent a sip service request
23:28:37: %LINK-3-UPDOWN: Interface GigabitEthernet2/1/1, changed state to up
23:28:38: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet2/1/1, changed
state to up
23:28:53: GE2/1/1 ACMGR: Rcv SIP msg: resp connect forwarded, hdl D6000002, sss_hdl 9E00000F

23:28:53: GE2/1/1 ACMGR: service connected event, SIP state chg connecting to connected,
action is respond forwarded
23:28:53: ACLIB: pthru_intf_response hdl is D6000002, response is 1
23:28:53: ACLIB [10.0.3.201, 5]: Setting new AC state to Ac-Connected
```

The command output is self-explanatory.

Related Commands

Command	Description
debug acircuit	Debugs errors and events that occur on the attachment circuits.

Command	Description
debug vpdn	Debugs errors and events relating to L2TP configuration and the surrounding Layer 2 tunneling infrastructure.
debug xconnect	Debugs errors and events related to an xconnect configuration.

debug l2vpn atom checkpoint

To enable the debugging of Any Transport over MPLS (AToM) events when AToM is configured for nonstop forwarding/stateful switchover (NSF/SSO) and graceful restart, use the **debug l2vpn atom checkpoint** command in privileged EXEC mode. To disable the debugging of these messages, use the **no** form of this command.

debug l2vpn atom checkpoint

no debug l2vpn atom checkpoint

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging of the AToM NSF/SSO and graceful restart is disabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.7S	This command was introduced as part of the Multiprotocol Label Switching (MPLS)-based Layer 2 VPN (L2VPN) command modifications for cross-OS support. This command will replace the **debug mpls l2transport checkpoint** command in future releases.
15.3(1)S	This command was integrated in Cisco IOS Release 15.3(1)S.

Usage Guidelines

Debug commands use a significant amount of CPU time and can affect system performance.

Examples

In the following example, the output shows that NSF/SSO and graceful restart synchronize the data between the active and backup Route Processors (RP) after an AToM virtual circuit (VC) is created. (Both **debug l2vpn atom checkpoint** and **debug l2vpn acircuit checkpoint** commands are enabled in this example.)

The **debug l2vpn atom checkpoint** command is enabled on the active RP:

```
Device# debug l2vpn atom checkpoint
Device# debug l2vpn acircuit checkpoint

AToM HA:
  AToM checkpointing events and errors debugging is on
AC HA:
  Attachment Circuit Checkpoint debugging is on

AToM HA [10.55.55.2, 1002]: Build provision msg, SSM sw/seg 8192/8194 [0x2000/0x2002] PW
id 9216 [0x2400] local label 21
AC HA: Dynamic Sync. Event:4 Sw:8192[2000] Se:16385[4001]
AToM HA: CF sync send complete
AC HA CF: Sync send complete. Code:0
```

On the standby RP, the following messages indicate that it receives checkpointing data:

```
AC HA [10.55.55.2, 1002]: Add to WaitQ. Flags:1
AToM HA [105.55.55.2, 1002]: Received 32-byte provision version 1 CF message
AC HA CF: ClientId:89, Entity:0 Length:40
AToM HA [10.55.55.2, 1002]: Process chkpt msg provision [1], ver 1
AToM HA [10.55.55.2, 1002]: Reserved SSM sw/seg 8192/8194 [0x2000/0x2002] PW id 9216 [0x2400]
AC HA: Process Msg:35586. Ptr:44CBFD90. Val:0
AC HA: Sync. Event:4 CktType:4 Sw:8192[2000] Se:16385[4001]
AC HA [10.55.55.2, 1002]: Remove from WaitQ. Flags:1[OK][OK]
```

During a switchover from the active to the backup RP, the following debug messages are displayed:

```
%HA-5-MODE: Operating mode is hsa, configured mode is sso.
AC HA RF: CId:83, Seq:710, Sta:RF_STATUS_OPER_REDUNDANCY_MODE_CHANGE, Opr:5, St:STANDBY
HOT, PSt:ACTIVE
AToM HA: CID 84, Seq 715, Status RF_STATUS_OPER_REDUNDANCY_MODE_CHANGE, Op 5, State STANDBY
 HOT, Peer ACTIVE
AC HA RF: CId:83, Seq:710, Sta:RF_STATUS_PEER_PRESENCE, Opr:0, St:STANDBY HOT, PSt:ACTIVE
AToM HA: CID 84, Seq 715, Status RF_STATUS_PEER_PRESENCE, Op 0, State STANDBY HOT, Peer
ACTIVE
AC HA RF: CId:83, Seq:710, Sta:RF_STATUS_PEER_COMM, Opr:0, St:STANDBY HOT, PSt:DISABLED
AToM HA: CID 84, Seq 715, Status RF_STATUS_PEER_COMM, Op 0, State STANDBY HOT, Peer DISABLED
%HA-2-CUTOVER_NOTICE: Cutover initiated. Cease all console activity until system restarts.
%HA-2-CUTOVER_NOTICE: Do not add/remove RSPs or line cards until switchover completes.
%HA-2-CUTOVER_NOTICE: Deinitializing subsystems...
%OIR-6-REMCARD: Card removed from slot 4, interfaces disabled
%OIR-6-REMCARD: Card removed from slot 5, interfaces disabled
%OIR-6-REMCARD: Card removed from slot 9, interfaces disabled
%HA-2-CUTOVER_NOTICE: Reinitializing subsystems...
%HA-2-CUTOVER_NOTICE: System preparing to restart...
%HA-5-NOTICE: Resuming initialization...
AC HA RF: CId:83, Seq:710, Sta:RF_STATUS_REDUNDANCY_MODE_CHANGE, Opr:7, St:STANDBY HOT,
PSt:DISABLED
.
.
.
%LDP-5-GR: LDP restarting gracefully.  Preserving forwarding state for 250 seconds.
AC HA RF: CId:83, Seq:710, Sta:RF_PROG_ACTIVE, Opr:0, St:ACTIVE, PSt:DISABLED
AToM HA: CID 84, Seq 715, Event RF_PROG_ACTIVE, Op 0, State ACTIVE, Peer DISABLED
AC HA: Process Msg:35588. Ptr:0. Val:0
AC HA: Switchover: Standby->Active
AC HA RF: Reconciling
```

Related Commands

Command	Description
debug l2vpn acircuit checkpoint	Enables the debugging of AToM attachment circuit events when AToM is configured for NSF/SSO and graceful restart.
debug mpls l2transport checkpoint	Enables the debugging of AToM events when AToM is configured for NSF/SSO and graceful restart.

debug l2vpn atom event-trace

To enable debugging of event trace information for Layer 2 VPN (L2VPN) Any Transport over MPLS (AToM), use the **debugl2vpn atom event-trace** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug l2vpn atom {**event-trace** *number* [**preserve**]}

no debug l2vpn atom {**event-trace** *number* [**preserve**]}

Syntax Description

number	Number of event trace logs to be stored per context.
preserve	(Optional) Specifies that the event trace logs should not be removed.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.7S	This command was introduced as part of the Multiprotocol Label Switching (MPLS)-based L2VPN command modifications for cross-OS support. This command will replace the **debug mpls l2transport event-trace** command in future releases.
15.3(1)S	This command was integrated in Cisco IOS Release 15.3(1)S.

Usage Guidelines

The **debug l2vpn atom event-trace** command does not produce any output of its own. Instead, it affects the size of the event-trace buffer for Any Transport over MPLS (AToM) events.

Examples

The following is sample output from the **debug l2vpn atom event-trace** command:

```
Device# debug l2vpn atom event-trace
AToM LDP event-trace debugging is on
```

Related Commands

Command	Description
debug mpls l2transport event-trace	Enables debugging of event trace information for MPLS Layer 2 transport events.

debug l2vpn atom fast-failure-detect

To enable the debugging of Layer 2 VPN (L2VPN) Any Transport over MPLS (AToM) fast failure detection, use the **debug l2vpn atom fast-failure-detect** command in privileged EXEC mode. To disable the debugging, use the **no** form of this command.

debug l2vpn atom fast-failure-detect

no debug l2vpn atom fast-failure-detect

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging of L2VPN fast failure detection is disabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.7S	This command was introduced as part of the Multiprotocol Label Switching (MPLS)-based L2VPN command modifications for cross-OS support. This command will replace the **debug mpls l2transport fast-failure-detect** command in future releases.
15.3(1)S	This command was integrated in Cisco IOS Release 15.3(1)S.

Examples

The following example shows how to enable L2VPN AToM fast failure detection:

```
Device# debug l2vpn atom fast-failure-detect

========= Line Card (Slot 3) =========
AToM fast failure detect debugging is on

00:03:28: AToM FFD[10.1.1.2]: Sending type: BFD, adjacency: DOWN, local: 10.1.1.1
00:03:28: AToM FFD[10.1.1.2]: ADJ_DOWN, local: 10.1.1.1
00:03:28: AToM FFD[10.1.1.2, 100]: ADJ_DOWN
```

Related Commands

Command	Description
debug mpls l2transport fast-failure-detection	Enables the debugging of fast failure detection.

debug l2vpn atom signaling

To enable debugging of Layer 2 VPN (L2VPN) Any Transport over MPLS (AToM) signaling protocol information, use the **debug l2vpn atom signaling** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug l2vpn atom signaling {**event**| **message**| **fsm**}

no debug l2vpn atom signaling {**event**| **message**| **fsm**}

Syntax Description

event	Enables debugging of protocol events.
message	Enables debugging of protocol messages.
fsm	Enables debugging of finite state machine (FSM).

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.7S	This command was introduced as part of the Multiprotocol Label Switching (MPLS)-based L2VPN command modifications for cross-OS support. This command will replace the **debug mpls l2transport signaling** command in future releases.
15.3(1)S	This command was integrated in Cisco IOS Release 15.3(1)S.

Examples

The following is sample output from the **debug l2vpn atom signaling** command:

```
Device# debug l2vpn atom signaling event
AToM LDP event debugging is on

Device# debug l2vpn atom signaling message
AToM LDP message debugging is on

AToM:
  AToM LDP event debugging is on
  AToM LDP message debugging is on
*Mar 24 23:10:55.611: AToM LDP [10.9.9.9]: Allocate LDP instance
*Mar 24 23:10:55.611: AToM LDP [10.9.9.9]: Opening session, 1 clients
*Mar 24 23:10:56.063: %SYS-5-CONFIG_I: Configured from console by console
*Mar 24 23:10:56.583: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial3/0, changed
state to up
*Mar 24 23:11:00.539: AToM LDP [10.9.9.9]: Session is up
*Mar 24 23:11:00.539: AToM LDP [10.9.9.9]: Peer address change, add 10.1.1.100
*Mar 24 23:11:00.539: AToM LDP [10.9.9.9]: Peer address change, add 10.1.1.6
*Mar 24 23:11:00.539: AToM LDP [10.9.9.9]: Peer address change, add 10.9.9.9
*Mar 24 23:11:00.539: AToM LDP [10.9.9.9]: Peer address change, add 10.1.1.6
```

```
*Mar 24 23:11:00.539: AToM LDP [10.9.9.9]: Sending label mapping msg
vc type 7, cbit 1, vc id 50, group id 6, vc label 21, status 0, mtu 1500
*Mar 24 23:11:00.539: AToM LDP [10.9.9.9]: Received label mapping msg, id 113
vc type 7, cbit 1, vc id 50, group id 6, vc label 21, status 0, mtu 1500
```

Related Commands

Command	Description
debug mpls l2transport signaling	Displays information about the AToM signaling protocol.

debug l2vpn atom static-oam

To enable the debugging of messages related to static operations administrative and management (OAM), use the **debug l2vpn atom static-oam** command in privileged EXEC mode. To disable the debugging of these messages, use the **no** form of this command.

debug l2vpn atom static-oam {**elog**| **error**| **event**| **fsm**}

no debug l2vpn atom static-oam [**elog**| **error**| **event**| **fsm**]

Syntax Description

elog	Displays logging messages for static pseudowire OAM.
error	Displays error messages for static pseudowire OAM.
event	Displays event messages for static pseudowire OAM.
fsm	Displays finite state machine (FSM) messages for static pseudowire OAM.

Command Default

Display of static pseudowire messages is not disabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.7S	This command was introduced as part of the Multiprotocol Label Switching (MPLS)-based Layer 2 VPN (L2VPN) command modifications for cross-OS support. This command will replace the **debug mpls l2transport static-oam** command in future releases.
15.3(1)S	This command was integrated in Cisco IOS Release 15.3(1)S.

Usage Guidelines

The **debug l2vpn atom static-oam error** does not produce any output of its own. Instead, it affects the size of the event-trace buffer for Any Transport over MPLS (AToM) events.

Examples

The following example enables the display of error messages for static pseudowire OAM:

```
Device# debug l2vpn atom static-oam error
Static PW OAM events debugging is on
```

Related Commands

Command	Description
debug mpls l2transport static-oam	Enables the debugging of messages related to static pseudowire operations OAM.
show l2vpn atom static-oam	Displays the status of static pseudowires.

debug l2vpn atom vc

To enable debugging of status of the Layer 2 VPN (L2VPN) Any Transport over MPLS (AToM) virtual circuits (VCs), use the **debug l2vpn atom vc** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug l2vpn atom vc {**event**| **fsm**| **ldp**| **subscriber**| **status** {**event**| **fsm**}}

no debug mpls l2transport vc {**event**| **fsm**| **ldp**| **subscriber**| **status** {**event**| **fsm**}}

Syntax Description

event	Displays AToM event messages about VCs.
fsm	Displays debug information related to the finite state machine (FSM).
ldp	Displays debug information related to the Label Distribution Protocol (LDP).
subscriber	Displays debug information related to the L2VPN subscriber.
status	Displays debug information related to the status of VCs.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.7S	This command was introduced as part of the Multiprotocol Label Switching (MPLS)-based L2VPN command modifications for cross-OS support. This command will replace the **debug mpls l2transport vc** command in future releases.
15.3(1)S	This command was integrated in Cisco IOS Release 15.3(1)S.

Usage Guidelines

You can issue this command from the line card or the Route Processor (RP).

Examples

The following is sample output from the **debug l2vpn atom vc event** and **debug l2vpn atom vc fsm** commands:

```
Device# debug l2vpn atom vc event
AToM vc event debugging is on

Device# debug l2vpn atom vc fsm
```

```
AToM vc fsm debugging is on

AToM:
  AToM vc event debugging is on
  AToM vc fsm debugging is on
*Mar 24 23:17:24.371: AToM MGR [10.9.9.9, 50]: Event provision, state changed from idle to
 provisioned
*Mar 24 23:17:24.371: AToM MGR [10.9.9.9, 50]: Provision vc
*Mar 24 23:17:24.371: AToM SMGR [10.9.9.9, 50]: Requesting VC create, vc_handle 61A09930
*Mar 24 23:17:24.371: AToM MGR [10.9.9.9, 50]: Event local up, state changed from provisioned
 to local standby
*Mar 24 23:17:24.371: AToM MGR [10.9.9.9, 50]: Update local vc label binding
*Mar 24 23:17:24.371: AToM SMGR [10.9.9.9, 50]: successfully processed create request
*Mar 24 23:17:24.875: %SYS-5-CONFIG_I: Configured from console by console
*Mar 24 23:17:25.131: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial3/0, changed
state to up
*Mar 24 23:17:28.567: AToM MGR [10.9.9.9, 50]: Event ldp up, state changed from local standby
 to local ready
*Mar 24 23:17:28.567: AToM MGR [10.9.9.9, 50]: Advertise local vc label binding
*Mar 24 23:17:28.567: AToM MGR [10.9.9.9, 50]: Event remote up, state changed from local
ready to establishing
*Mar 24 23:17:28.567: AToM MGR [10.9.9.9, 50]: Remote end up
*Mar 24 23:17:28.567: AToM MGR [10.9.9.9, 50]: Event remote validated, state changed from
establishing to established
*Mar 24 23:17:28.567: AToM MGR [10.9.9.9, 50]: Validate vc, activating data plane
*Mar 24 23:17:28.567: AToM SMGR [10.9.9.9, 50]: Processing imposition update, vc_handle
61A09930, update_action 3, remote_vc_label 21
*Mar 24 23:17:28.567: AToM SMGR [10.9.9.9, 50]: Imposition Programmed, Output Interface:
PO5/0
*Mar 24 23:17:28.567: AToM SMGR [10.9.9.9, 50]: Processing disposition update, vc_handle
61A09930, update_action 3, local_vc_label 22
*Mar 24 23:17:28.571: AToM SMGR: Processing TFIB event for 10.9.9.9
*Mar 24 23:17:28.571: AToM SMGR [10.9.9.9, 50]: Imposition Programmed, Output Interface:
PO5/0
```

The following is sample output of MPLS pseudowire status signaling messages from the **debug l2vpn atom vc status event** and **debug l2vpn atom vc status fsm** commands:

```
Device# debug l2vpn atom vc status event
Device# debug l2vpn atom vc status fsm

*Feb 26 14:03:42.543: AToM MGR [10.9.9.9, 100]: Receive SSS STATUS(UP)
*Feb 26 14:03:42.543: AToM MGR [10.9.9.9, 100]: AC status UP
*Feb 26 14:03:42.543: AToM MGR [10.9.9.9, 100]: S:Evt local up, LndRru->LnuRru
*Feb 26 14:03:42.543: AToM MGR [10.9.9.9, 100]: S:Evt local ready, LnuRru->LruRru
*Feb 26 14:03:42.543: AToM MGR [10.9.9.9, 100]: S:Act send label(UP)
*Feb 26 14:03:42.543: AToM MGR [10.9.9.9, 100]: Send label(UP)
*Feb 26 14:03:42.543: AToM MGR [10.9.9.9, 100]: Local AC : UP
*Feb 26 14:03:42.543: AToM MGR [10.9.9.9, 100]: Dataplane: no fault
*Feb 26 14:03:42.543: AToM MGR [10.9.9.9, 100]: Overall  : no fault
*Feb 26 14:03:42.543: AToM MGR [10.9.9.9, 100]: Remote label is ready
*Feb 26 14:03:42.543: AToM MGR [10.9.9.9, 100]: S:Evt remote ready in LruRru
*Feb 26 14:03:42.543: AToM MGR [10.9.9.9, 100]: S:Evt remote up in LruRru
*Feb 26 14:03:42.543: AToM MGR [10.9.9.9, 100]: S:Evt dataplane clear fault in LruRru
*Feb 26 14:03:42.543: AToM MGR [10.9.9.9, 100]: S:Evt dataplane clear fault in LruRru
*Feb 26 14:03:42.551: AToM MGR [10.9.9.9, 100]: S:Evt dataplane clear fault in LruRru
```

The status codes in the messages, such as S and LruRru, indicate the status of the local and remote devices. The following is the list status codes displayed in the output:

- L—local router
- R—remote router
- r or n—ready (r) or not ready (n)
- u or d—up (u) or down (d) status

The output also includes the following values:

- D—Dataplane
- S—Local shutdown

Related Commands

Command	Description
debug mpls l2transport vc	Enables debugging of the AToM VCs.

debug l2vpn atom vc vccv

To enable Layer 2 VPN (L2VPN) Any Transport over MPLS (AToM) Virtual Circuit Connection Verification (VCCV) debugging, use the **debug l2vpn atom vc vccv** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug l2vpn atom vc vccv [bfd] event

no debug l2vpn atom vc vccv [bfd] event

Syntax Description

bfd	(Optional) Displays event messages when Bidirectional Forwarding Detection (BFD) sessions are created, when BFD sends dataplane fault notifications to L2VPN, and when L2VPN sends the attachment circuit (AC) signaling status to BFD.
event	Displays AToM event messages about the VCCV.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.7S	This command was introduced as part of the Multiprotocol Label Switching (MPLS)-based L2VPN command modifications for cross-OS support. This command will replace the **debug mpls l2transport vc vccv** command in future releases.
15.3(1)S	This command was integrated in Cisco IOS Release 15.3(1)S.

Usage Guidelines

Use this command to enable L2VPN AToM VCCV events and AToM VCCV BFD events debugging.

Examples

The following example shows how to enable MPLS L2VPN virtual circuit (VC) VCCV BFD event debugging:

```
Device# debug l2vpn atom vc vccv bfd event
AToM VCCV BFD events debugging is on

Aug 10 16:55:41.492: AToM VCCV BFD[10.1.1.2, 1234000]: ......... context not found
*Aug 10 16:55:41.492: AToM VCCV BFD[10.1.1.2, 1234000]: ......... context not found
*Aug 10 16:55:41.492: AToM VCCV BFD[10.1.1.2, 1234000]: ............ context not found
*Aug 10 16:55:41.492: AToM VCCV BFD[10.1.1.2, 1234000]: ....... context not found
*Aug 10 16:55:41.493: AToM VCCV BFD[10.1.1.2, 1234000]: .. Session create
*Aug 10 16:55:41.493: AToM VCCV BFD[10.1.1.2, 1234000]: ..  next-hop         2.1.1.2:1
*Aug 10 16:55:41.493: AToM VCCV BFD[10.1.1.2, 1234000]: ..  cc_type          1
*Aug 10 16:55:41.493: AToM VCCV BFD[10.1.1.2, 1234000]: ..  cv_type          5
*Aug 10 16:55:41.493: AToM VCCV BFD[10.1.1.2, 1234000]: ..  CC control word  enabled
```

```
*Aug 10 16:55:41.493: AToM VCCV BFD[10.1.1.2, 1234000]: ..  CV Fault Detection and Signaling
 without IP/UDP headers
*Aug 10 16:55:41.500: AToM VCCV BFD[10.1.1.2, 1234000]: .. create 00000001/2A98A72F40
*Aug 10 16:55:41.500: AToM VCCV BFD[10.1.1.2, 1234000]: .. lookup added 00000001
*Aug 10 16:55:42.315: AToM VCCV BFD[10.1.1.2, 1234000]: session 00000001 ADJ UP
*Aug 10 16:55:42.315: AToM VCCV BFD[10.1.1.2, 1234000]: inform BFD, status UP, event 1
*Aug 10 16:55:42.315: AToM VCCV BFD[10.1.1.2, 1234000]: Start VCCV BFD status timer
*Aug 10 16:55:45.374: AToM VCCV BFD[10.1.1.2, 1234000]: VCCV BFD status timer expired
*Aug 10 16:55:45.374: AToM VCCV BFD[10.1.1.2, 1234000]: session 00000001 BFD STATUS UP
```

Related Commands

Command	Description
debug mpls l2transport vc vccv	Enables AToM VCCV debugging
show mpls l2transport vc	Displays information about the status of the AToM VCs.

debug l2vpn pseudowire

To enable debugging information for Layer 2 VPN (L2VPN) pseudowire configuration, use the **debug l2vpn pseudowire** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug l2vpn pseudowire {**event**| **error**}

no debug l2vpn pseudowire {**event**| **error**}

Syntax Description

event	Displays debugging information for L2VPN pseudowire events.
error	Displays debugging information for L2VPN pseudowire errors.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.7S	This command was introduced as part of the Multiprotocol Label Switching (MPLS)-based L2VPN command modifications for cross-OS support.
15.3(1)S	This command was integrated into Cisco IOS release 15.3(1)S.

Examples

The following is sample output from the **debug l2vpn pseudowire event** command:

```
Device# debug l2vpn pseudowire event
L2VPN pseudowire events debugging is on.

*Aug 10 17:52:22.896: Pseudowire[ps1]: PW interface not yet in a L2VPN service, ignore
[no]shutdown
*Aug 10 17:52:25.851: Pseudowire[pw1]: Pseudowire interface: peer id 10.0.0.0 not configured

*Aug 10 17:52:25.851: Pseudowire[pw1]: Pseudowire interface config still incomplete, skip
update to xconnect db
*Aug 10 17:52:33.727: PWCFG WAVL Event: Updating pwid: 10002 peer: 10.1.1.2 vcid: 1234000
*Aug 10 17:52:33.727: PWCFG WAVL Event: pwid: 10002 alloc peer 10.1.1.2 vcid: 1234000
*Aug 10 17:52:33.727: Pseudowire[pw1]: Pseudowire interface not yet associated with a L2VPN
 service
```

debug l2vpn vfi

To enable debugging layer 2 VPN (L2VPN) virtual forwarding instance (VFI) events and errors, use the **debug l2vpn vfi** command in privileged EXEC mode. To disable debugging of VFI events and errors, use the **no** form of this command.

debug l2vpn vfi [**fsm**] {**error**| **event**}

no debug l2vpn vfi [**fsm**] {**error**| **event**}

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.7S	This command was introduced. This command will replace the **debug vfi** command in future releases.
15.3(1)S	This command was integrated in Cisco IOS Release 15.3(1)S.

Examples

The following is sample output from the **debug l2vpn vfi** command:

```
Device# debug l2vpn vfi
```

Related Commands

Command	Description
debug vfi	Enables debugging VFI events and errors.

debug l2vpn xconnect

To enable the debugging information about a Layer 2 VPN (L2VPN) xconnect configuration, use the **debug l2vpn xconnect** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug l2vpn xconnect {**error**| **event** [**detail**]| **initialization**| **internal**| **monitor**}

no debug l2vpn xconnect {**error**| **event** [**detail**]| **initialization**| **internal**| **monitor**}

Syntax Description

error	Displays errors related to an xconnect configuration.
event	Displays events related to an xconnect configuration.
detail	(Optional) Displays the xconnect detailed debugging information.
initialization	Displays information about xconnect initialization events.
internal	Displays information about xconnect internal events.
monitor	Displays debugging information about xconnect peer monitoring debugs.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.7S	This command was introduced as part of the Multiprotocol Label Switching (MPLS)-based L2VPN command modifications for cross-OS support. This command will replace the **debug xconnect** command in future releases.
15.3(1)S	This command was integrated in Cisco IOS Release 15.3(1)S.

Examples

The following is sample output from the **debug l2vpn xconnect** command for an xconnect session on a Gigabit Ethernet interface:

```
Device# debug l2vpn xconnect event

00:01:16: XC AUTH [Gi2/1/1, 5]: Event: start xconnect authorization, state changed from
IDLE to AUTHORIZING
00:01:16: XC AUTH [Gi2/1/1, 5]: Event: found xconnect authorization, state changed from
AUTHORIZING to DONE
00:01:16: XC AUTH [Gi2/1/1, 5]: Event: free xconnect authorization request, state changed
from DONE to END
```

Related Commands

Command	Description
debug xconnect	Enables the debugging information for an xconnect configuration.

debug l3-mgr tunnel

To enable debugging for interface, tunnel, or VLAN events for the Layer 3 manager infrastructure on RP of Cisco 7600 routers, use the **debug l3-mgr tunnel** command. To disable the debugging, use the **no** form of the command.

debug l3-mgr tunnel

no debug l3-mgr tunnel

Syntax Description

l3-mgr	Displays debugging output for the Layer 3 manager infrastructure on Cisco 7600 routers.
tunnel	Displays all tunnel related reserved VLAN events.

Command Default

None

Command Modes

Privileged EXEC

Command History

Release	Modification
15.3(2)S	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Use the debug command only to troubleshoot specific problems, or during troubleshooting sessions with Cisco technical support staff.

Examples

The following shows sample output for the **debug l3-mgr tunnel** command:

```
CE1#debug l3-mgr tunnel
l3 mgr tunnel debugging is on
*Mar  1 09:50:53.431 IST: l3mgr_tunnel_checking_src_address:
Tunnel[Tunnel56] src[64003801] tbl_id[0] state changed to DOWN
*Mar  1 09:50:53.431 IST: l3mgr_tunnel_checking_src_address:
Checked Tunnel[Tunnel110] src[64006E01] if_up[UP] tbl_id[0]
*Mar  1 09:50:53.431 IST: l3mgr_tunnel_checking_src_address:
Checked Tunnel[Tunnel109] src[64006D01] if_up[UP] tbl_id[0]
*Mar  1 09:50:53.431 IST: l3mgr_tunnel_checking_src_address:
Checked Tunnel[Tunnel108] src[64006C01] if_up[UP] tbl_id[0]
*Mar  1 09:50:53.4no sh31 IST: l3mgr_tunnel_checking_src_address:
Checked Tunnel[Tunnel107] src[64006B01] if_up[UP] tbl_id[0]
*Mar  1 09:50:53.431 IST: l3mgr_tunnel_checking_src_address:
Checked Tunnel[Tunnel106] src[64006A01] if_up[UP] tbl_id[0]
*Mar  1 09:50:53.431 IST: l3mgr_tunnel_checking_src_address:
Checked Tunnel[Tunnel105] src[64006901] if_up[UP] tbl_id[0]
*Mar  1 09:50:53.431 IST: l3mgr_tunnel_checking_src_address:
Checked Tunnel[Tunnel104] src[64006801] if_up[UP] tbl_id[0]
```

```
*Mar  1 09:50:53.431 IST: l3mgr_tunnel_checking_src_address:
Checked Tunnel[Tunnel103] src[64006701] if_up[UP] tbl_id[0]
*Mar  1 09:50:53.431 IST: l3mgr_tunnel_checking_src_address:
Checked Tunnel[Tunnel102] src[64006601] if_up[UP] tbl_id[0]
*Mar  1 09:50:53.435 IST: l3mgr_tunnel_checking_src_address:
Checked Tunnel[Tunnel101] src[64006501] if_up[UP] tbl_id[0]
*Mar  1 09:50:53.435 IST: l3mgr_tunnel_checking_src_address:
Checked Tunnel[Tunnel100] src[64006401] if_up[UP] tbl_id[0]
```

debug l4f

To enable troubleshooting for Layer 4 Forwarding (L4F) flows, use the **debug l4f** command in privileged EXEC mode. To disable the troubleshooting, use the **no** form of this command.

debug l4f {**api**| **flow-db**| **flows**| **packet** {**all**| **detail**| **injection**| **interception**| **proxying**| **spoofing**}| **test-app**| **trace-db-api**| **trace-db-flow**| **trace-engine**}

no debug l4f {**api**| **flow-db**| **flows**| **packet** {**all**| **detail**| **injection**| **interception**| **proxying**| **spoofing**}| **test-app**| **trace-db-api**| **trace-db-flow**| **trace-engine**}

Syntax Description

api	Toggles L4F API debugging.
flow-db	Toggles L4F flow database debugging.
flows	Toggles L4F flows debugging.
packet	Toggles L4F packet debugging.
all	Toggles all L4F packet debugging.
detail	Toggles L4F packet detail debugging.
injection	Toggles L4F packet injection debugging.
interception	Toggles L4F packet interception debugging.
proxying	Toggles L4F packet proxying debugging.
spoofing	Toggles L4F packet spoofing debugging.
test-app	Toggles L4F test application debugging.
trace-db-api	Toggles L4F database API debugging.
trace-db-flow	Toggles L4F database flow debugging.
trace-engine	Toggles L4F API tracing debugging.

Command Default

L4F debugging is off.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.1(2)T	This command was introduced.

Usage Guidelines

Use this command to enable debugging for Layer 4 forwarding flows.

Examples

The following example shows how to enable debugging for L4F packets:

```
Router# debug l4f packet all
```

Related Commands

Command	Description
show l4f	Displays the flow database for L4F.

debug lacp

To enable debugging of all Link Aggregation Control Protocol (LACP) activity, use the **debug lacp**command in privileged EXEC mode. To disable LACP debugging, use the **no** form of this command.

debug lacp [**all**| **event**| **fsm**| **misc**| **multi-chassis** [**all**| **database**| **lacp-mgr**| **redundancy-group**| **user-interface**]| **packet**]

no debug lacp [**all**| **event**| **fsm**| **misc**| **multi-chassis** [**all**| **database**| **lacp-mgr**| **redundancy-group**| **user-interface**]| **packet**]

Syntax Description

all	(Optional) Activates debugging for all LACP operations.
event	(Optional) Activates debugging of events that occur within LACP.
fsm	(Optional) Activates debugging for changes within the LACP finite state machine.
misc	(Optional) Activates debugging for various operations that may be useful for monitoring the status of LACP.
multi-chassis	(Optional) Activates multi-chassis LACP (mLACP) debugging.
all	(Optional) Activates all mLACP debugging.
database	(Optional) Activates mLACP database debugging.
lacp-mgr	(Optional) Activates mLACP interface debugging.
redundancy-group	(Optional) Activates mLACP interchassis redundancy group debugging.
user-interface	(Optional) Activates mLACP interchassis user interface debugging.
packet	(Optional) Displays the receiving and transmitting LACP control packets.

Command Default

LACP debugging activity is disabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.1(13)EW	Support for this command was introduced on the Cisco Catalyst 4500 series switch.
12.2(31)SB2	This command was integrated into Cisco IOS Release 12.2(31)SB.
12.2(33)SRB	Support for this command on the Cisco 7600 router was integrated into Cisco IOS Release 12.2(33)SRB.
Cisco IOS XE Release 2.4	This command was integrated into Cisco IOS XE Release 2.4.
12.2(33)SRE	This command was modified. The following keywords were added: **multi-chassis**, **all**, **database**, **lacp-mgr**, **redundancy-group**, and **user-interface**.

Usage Guidelines

This command is useful for troubleshooting problems with LACP.

Examples

The following sample output from the **debug lacp all** command shows LACP activity on a port-channel member link Gigabit Ethernet 5/0/0:

```
Router# debug lacp all
Link Aggregation Control Protocol all debugging is on
Router1#
*Aug 20 17:21:51.685: LACP :lacp_bugpak: Receive LACP-PDU packet via Gi5/0/0
*Aug 20 17:21:51.685: LACP : packet size: 124
*Aug 20 17:21:51.685: LACP: pdu: subtype: 1, version: 1
*Aug 20 17:21:51.685: LACP: Act: tlv:1, tlv-len:20, key:0x1, p-pri:0x8000, p:0x14,
p-state:0x3C,
s-pri:0xFFFF, s-mac:0011.2026.7300
*Aug 20 17:21:51.685: LACP: Part: tlv:2, tlv-len:20, key:0x5, p-pri:0x8000, p:0x42,
p-state:0x3D,
s-pri:0x8000, s-mac:0014.a93d.4a00
*Aug 20 17:21:51.685: LACP: col-tlv:3, col-tlv-len:16, col-max-d:0x8000
*Aug 20 17:21:51.685: LACP: term-tlv:0 termr-tlv-len:0
*Aug 20 17:21:51.685: LACP: Gi5/0/0 LACP packet received, processing
*Aug 20 17:21:51.685:     lacp_rx Gi5: during state CURRENT, got event 5(recv_lacpdu)
*Aug 20 17:21:59.869: LACP: lacp_p(Gi5/0/0) timer stopped
*Aug 20 17:21:59.869: LACP: lacp_p(Gi5/0/0) expired
*Aug 20 17:21:59.869:     lacp_ptx Gi5: during state SLOW_PERIODIC, got event 3(pt_expired)
*Aug 20 17:21:59.869: @@@ lacp_ptx Gi5: SLOW_PERIODIC -> PERIODIC_TX
*Aug 20 17:21:59.869: LACP: Gi5/0/0 lacp_action_ptx_slow_periodic_exit entered
*Aug 20 17:21:59.869: LACP: lacp_p(Gi5/0/0) timer stopped
*Aug 20 17:22:00.869: LACP: lacp_t(Gi5/0/0) timer stopped
*Aug 20 17:22:00.869: LACP: lacp_t(Gi5/0/0) expired
*Aug 20 17:22:19.089: LACP :lacp_bugpak: Receive LACP-PDU packet via Gi5/0/0
*Aug 20 17:22:19.089: LACP : packet size: 124
*Aug 20 17:22:19.089: LACP: pdu: subtype: 1, version: 1
*Aug 20 17:22:19.089: LACP: Act: tlv:1, tlv-len:20, key:0x1, p-pri:0x8000, p:0x14,
p-state:0x4,
s-pri:0xFFFF, s-mac:0011.2026.7300
*Aug 20 17:22:19.089: LACP: Part: tlv:2, tlv-len:20, key:0x5, p-pri:0x8000, p:0x42,
p-state:0x34,
s-pri:0x8000, s-mac:0014.a93d.4a00
*Aug 20 17:22:19.089: LACP: col-tlv:3, col-tlv-len:16, col-max-d:0x8000
*Aug 20 17:22:19.089: LACP: term-tlv:0 termr-tlv-len:0
*Aug 20 17:22:19.089: LACP: Gi5/0/0 LACP packet received, processing
```

```
*Aug 20 17:22:19.089:     lacp_rx Gi5: during state CURRENT, got event 5(recv_lacpdu)
*Aug 20 17:22:19.989: LACP: lacp_t(Gi5/0/0) timer stopped
*Aug 20 17:22:19.989: LACP: lacp_t(Gi5/0/0) expired
*Aug 20 17:22:19.989: LACP: timer lacp_t(Gi5/0/0) started with interval 1000.
*Aug 20 17:22:19.989: LACP: lacp_send_lacpdu: (Gi5/0/0) About to send the 110 LACPDU
*Aug 20 17:22:19.989: LACP :lacp_bugpak: Send LACP-PDU packet via Gi5/0/0
*Aug 20 17:22:19.989: LACP : packet size: 124
*Aug 20 17:22:20.957: LACP: lacp_t(Gi5/0/0) timer stopped
*Aug 20 17:22:20.957: LACP: lacp_t(Gi5/0/0) expired
*Aug 20 17:22:21.205: %LINK-3-UPDOWN: Interface GigabitEthernet5/0/0, changed state to down
*Aug 20 17:22:21.205: LACP: lacp_hw_off: Gi5/0/0 is going down
*Aug 20 17:22:21.205: LACP: if_down: Gi5/0/0
*Aug 20 17:22:21.205:    lacp_ptx Gi5: during state SLOW_PERIODIC, got event 0(no_periodic)
*Aug 20 17:22:22.089: %LINEPROTO-5-UPDOWN: Line protocol on Interface Port-channel5, changed
 state to down
*Aug 20 17:22:22.153: %C10K_ALARM-6-INFO: CLEAR CRITICAL GigE 5/0/0 Physical Port Link Down

*Aug 20 17:22:23.413: LACP: Gi5/0/0 oper-key: 0x0
*Aug 20 17:22:23.413: LACP: lacp_hw_on: Gi5/0/0 is coming up
*Aug 20 17:22:23.413:     lacp_ptx Gi5: during state NO_PERIODIC, got event 0(no_periodic)
*Aug 20 17:22:23.413: @@@ lacp_ptx Gi5: NO_PERIODIC -> NO_PERIODIC
*Aug 20 17:22:23.413: LACP: Gi5/0/0 lacp_action_ptx_no_periodic entered
*Aug 20 17:22:23.413: LACP: lacp_p(Gi5/0/0) timer stopped
*Aug 20 17:22:24.153: %LINK-3-UPDOWN: Interface GigabitEthernet5/0/0, changed state to up
*Aug 20 17:22:24.153: LACP: lacp_hw_on: Gi5/0/0 is coming up
*Aug 20 17:22:24.153:    lacp_ptx Gi5: during state FAST_PERIODIC, got event 0(no_periodic)
*Aug 20 17:22:24.153: @@@ lacp_ptx Gi5: FAST_PERIODIC -> NO_PERIODIC
*Aug 20 17:22:24.153: LACP: Gi5/0/0 lacp_action_ptx_fast_periodic_exit entered
*Aug 20 17:22:24.153: LACP: lacp_p(Gi5/0/0) timer stopped
*Aug 20 17:22:24.153: LACP:
*Aug 20 17:22:25.021: LACP: lacp_p(Gi5/0/0) timer stopped
*Aug 20 17:22:25.021: LACP: lacp_p(Gi5/0/0) expired
*Aug 20 17:22:25.021:    lacp_ptx Gi5: during state FAST_PERIODIC, got event 3(pt_expired)
*Aug 20 17:22:25.021: @@@ lacp_ptx Gi5: FAST_PERIODIC -> PERIODIC_TX
*Aug 20 17:22:25.021: LACP: Gi5/0/0 lacp_action_ptx_fast_periodic_exit entered
*Aug 20 17:22:25.021: LACP: lacp_p(Gi5/0/0) timer stopped
*Aug 20 17:22:25.917: LACP: lacp_p(Gi5/0/0) timer stopped
*Aug 20 17:22:25.917: LACP: lacp_p(Gi5/0/0) expired
*Aug 20 17:22:25.917:    lacp_ptx Gi5: during state FAST_PERIODIC, got event 3(pt_expired)
*Aug 20 17:22:25.917: @@@ lacp_ptx Gi5: FAST_PERIODIC -> PERIODIC_TX
*Aug 20 17:22:25.917: LACP: Gi5/0/0 lacp_action_ptx_fast_periodic_exit entered
*Aug 20 17:22:25.917: LACP: lacp_p(Gi5/0/0) timer stopped
Router1#
```

debug lane client

Note Effective with Cisco IOS Release 15.1M, the **debug lane client** command is not available in Cisco IOS software.

To display information about a LAN Emulation Client (LEC), use the **debug lane client** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug lane client {**all**| **le-arp**| **mpoa**| **packet**| **signaling**| **state**| **topology**} [**interface** *interface*]

no debug lane client {**all**| **le-arp**| **mpoa**| **packet**| **signaling**| **state**| **topology**} [**interface** *interface*]

Syntax Description

all	Displays all debug information related to the LEC.
le-arp	Displays debug information related to the LAN Emulation (LANE) Address Resolution Protocol (ARP) table.
mpoa	Displays debug information to track the following: • MPOA specific TLV information in le-arp requests/responses • Elan-id and local segment TLV in lane control frames • When a LANE client is bound to an MPC/MPS
packet	Displays debug information about each packet.
signaling	Displays debug information related to client switched virtual circuits (SVCs).
state	Displays debug information when the state changes.
topology	Displays debug information related to the topology of the emulated LAN (ELAN).
interface *interface*	(Optional) Limits the debugging output to messages that relate to a particular interface or subinterface. If you enter this command multiple times with different interfaces, the last interface entered will be the one used to filter the messages.

Command Default

If the interface number is not specified, the default will be the number of all the **mpoa lane** clients.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.0(1)T	This command was introduced.
15.1M	This command was removed.

Usage Guidelines

The **debug lane client all** command can generate a large amount of output. Use a limiting keyword or specify a subinterface to decrease the amount of output and focus on the information you need.

Examples

The following example shows output for **debug lane client packet** and **debug lane client state** commands for an LEC joining an ELAN named elan1:

```
Router# debug lane client packet
Router# debug lane client state
```

The LEC listens for signaling calls to its ATM address (Initial State):

```
LEC ATM2/0.1: sending LISTEN
LEC ATM2/0.1:   listen on       39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1: received LISTEN
```

The LEC calls the LAN Emulation Configuration Server (LECS) and attempts to set up the Configure Direct VC (LECS Connect Phase):

```
LEC ATM2/0.1: sending SETUP
LEC ATM2/0.1:   callid          0x6114D174
LEC ATM2/0.1:   called party    39.020304050607080910111213.00000CA05B43.00
LEC ATM2/0.1:   calling_party   39.020304050607080910111213.00000CA05B40.01
```

The LEC receives a CONNECT response from the LECS. The Configure Direct VC is established:

```
LEC ATM2/0.1: received CONNECT
LEC ATM2/0.1:   callid          0x6114D174
LEC ATM2/0.1:   vcd             148
```

The LEC sends a CONFIG REQUEST to the LECS on the Configure Direct VC (Configuration Phase):

```
LEC ATM2/0.1: sending LANE_CONFIG_REQ on VCD 148
LEC ATM2/0.1:   SRC MAC address 0000.0ca0.5b40
LEC ATM2/0.1:   SRC ATM address 39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:   LAN Type        2
LEC ATM2/0.1:   Frame size      2
LEC ATM2/0.1:   LAN Name        elan1
LEC ATM2/0.1:   LAN Name size   5
```

The LEC receives a CONFIG RESPONSE from the LECS on the Configure Direct VC:

```
LEC ATM2/0.1: received LANE_CONFIG_RSP on VCD 148
LEC ATM2/0.1:   SRC MAC address 0000.0ca0.5b40
LEC ATM2/0.1:   SRC ATM address 39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:   LAN Type        2
LEC ATM2/0.1:   Frame size      2
```

```
LEC ATM2/0.1:   LAN Name        elan1
LEC ATM2/0.1:   LAN Name size   5
```

The LEC releases the Configure Direct VC:

```
LEC ATM2/0.1: sending RELEASE
LEC ATM2/0.1:   callid          0x6114D174
LEC ATM2/0.1:   cause code      31
```

The LEC receives a RELEASE_COMPLETE from the LECS:

```
LEC ATM2/0.1: received RELEASE_COMPLETE
LEC ATM2/0.1:   callid          0x6114D174
LEC ATM2/0.1:   cause code      16
```

The LEC calls the LAN Emulation Server (LES) and attempts to set up the Control Direct VC (Join/Registration Phase):

```
LEC ATM2/0.1: sending SETUP
LEC ATM2/0.1:   callid          0x61167110
LEC ATM2/0.1:   called party    39.020304050607080910111213.00000CA05B41.01
LEC ATM2/0.1:   calling_party   39.020304050607080910111213.00000CA05B40.01
```

The LEC receives a CONNECT response from the LES. The Control Direct VC is established:

```
LEC ATM2/0.1: received CONNECT
LEC ATM2/0.1:   callid          0x61167110
LEC ATM2/0.1:   vcd             150
```

The LEC sends a JOIN REQUEST to the LES on the Control Direct VC:

```
LEC ATM2/0.1: sending LANE_JOIN_REQ on VCD 150
LEC ATM2/0.1:   Status          0
LEC ATM2/0.1:   LECID           0
LEC ATM2/0.1:   SRC MAC address 0000.0ca0.5b40
LEC ATM2/0.1:   SRC ATM address 39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:   LAN Type        2
LEC ATM2/0.1:   Frame size      2
LEC ATM2/0.1:   LAN Name        elan1
LEC ATM2/0.1:   LAN Name size   5
```

The LEC receives a SETUP request from the LES to set up the Control Distribute VC:

```
LEC ATM2/0.1: received SETUP
LEC ATM2/0.1:   callid          0x6114D174
LEC ATM2/0.1:   called party    39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:   calling_party   39.020304050607080910111213.00000CA05B41.01
```

The LEC responds to the LES call setup with a CONNECT:

```
LEC ATM2/0.1: sending CONNECT
LEC ATM2/0.1:   callid          0x6114D174
LEC ATM2/0.1:   vcd             151
```

A CONNECT_ACK is received from the ATM switch. The Control Distribute VC is established:

```
LEC ATM2/0.1: received CONNECT_ACK
The LEC receives a JOIN response from the LES on the Control Direct VC.
LEC ATM2/0.1: received LANE_JOIN_RSP on VCD 150
LEC ATM2/0.1:   Status          0
LEC ATM2/0.1:   LECID           1
LEC ATM2/0.1:   SRC MAC address 0000.0ca0.5b40
LEC ATM2/0.1:   SRC ATM address 39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:   LAN Type        2
LEC ATM2/0.1:   Frame size      2
LEC ATM2/0.1:   LAN Name        elan1
LEC ATM2/0.1:   LAN Name size   5
```

The LEC sends an LE ARP request to the LES to obtain the broadcast and unknown server (BUS) ATM NSAP address (BUS connect):

```
LEC ATM2/0.1: sending LANE_ARP_REQ on VCD 150
```

```
LEC ATM2/0.1:   SRC MAC address     0000.0ca0.5b40
LEC ATM2/0.1:   SRC ATM address     39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:   TARGET MAC address     ffff.ffff.ffff
LEC ATM2/0.1:   TARGET ATM address  00.000000000000000000000000.000000000000.00
```

The LEC receives its own LE ARP request via the LES over the Control Distribute VC:

```
LEC ATM2/0.1: received LANE_ARP_RSP on VCD 151
LEC ATM2/0.1:   SRC MAC address     0000.0ca0.5b40
LEC ATM2/0.1:   SRC ATM address     39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:   TARGET MAC address     ffff.ffff.ffff
LEC ATM2/0.1:   TARGET ATM address  39.020304050607080910111213.00000CA05B42.01
```

The LEC calls the BUS and attempts to set up the Multicast Send VC:

```
LEC ATM2/0.1: sending SETUP
LEC ATM2/0.1:   callid          0x6114D354
LEC ATM2/0.1:   called party    39.020304050607080910111213.00000CA05B42.01
LEC ATM2/0.1:   calling_party   39.020304050607080910111213.00000CA05B40.01
```

The LEC receives a CONNECT response from the BUS. The Multicast Send VC is established:

```
LEC ATM2/0.1: received CONNECT
LEC ATM2/0.1:   callid          0x6114D354
LEC ATM2/0.1:   vcd             153
```

The LEC receives a SETUP request from the BUS to set up the Multicast Forward VC:

```
LEC ATM2/0.1: received SETUP
LEC ATM2/0.1:   callid          0x610D4230
LEC ATM2/0.1:   called party    39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:   calling_party   39.020304050607080910111213.00000CA05B42.01
```

The LEC responds to the BUS call setup with a CONNECT:

```
LEC ATM2/0.1: sending CONNECT
LEC ATM2/0.1:   callid          0x610D4230
LEC ATM2/0.1:   vcd             154
```

A CONNECT_ACK is received from the ATM switch. The Multicast Forward VC is established:

```
LEC ATM2/0.1: received CONNECT_ACK
The LEC moves into the OPERATIONAL state.
%LANE-5-UPDOWN: ATM2/0.1 elan elan1: LE Client changed state to up
```

The following output is from the **show lane client** command after the LEC joins the emulated LAN as shown in the **debug lane client** output:

```
Router# show lane client
LE Client ATM2/0.1  ELAN name: elan1  Admin: up  State: operational
Client ID: 1                LEC up for 1 minute 2 seconds
Join Attempt: 1
HW Address: 0000.0ca0.5b40   Type: token ring          Max Frame Size: 4544
Ring:1      Bridge:1        ELAN Segment ID: 2048
ATM Address: 39.020304050607080910111213.00000CA05B40.01
 VCD  rxFrames  txFrames  Type       ATM Address
   0         0         0  configure  39.020304050607080910111213.00000CA05B43.00
 142         1         2  direct     39.020304050607080910111213.00000CA05B41.01
 143         1         0  distribute 39.020304050607080910111213.00000CA05B41.01
 145         0         0  send       39.020304050607080910111213.00000CA05B42.01
 146         1         0  forward    39.020304050607080910111213.00000CA05B42.01
```

The following example shows **debug lane client** all command output when an interface with LECS, an LES/BUS, and an LEC is shut down:

```
Router# debug lane client all
LEC ATM1/0.2: received RELEASE_COMPLETE
LEC ATM1/0.2:   callid          0x60E8B474
LEC ATM1/0.2:   cause code      0
LEC ATM1/0.2: action A_PROCESS_REL_COMP
LEC ATM1/0.2: action A_TEARDOWN_LEC
LEC ATM1/0.2: sending RELEASE
```

```
LEC ATM1/0.2:   callid          0x60EB6160
LEC ATM1/0.2:   cause code      31
LEC ATM1/0.2: sending RELEASE
LEC ATM1/0.2:   callid          0x60EB7548
LEC ATM1/0.2:   cause code      31
LEC ATM1/0.2: sending RELEASE
LEC ATM1/0.2:   callid          0x60EB9E48
LEC ATM1/0.2:   cause code      31
LEC ATM1/0.2: sending CANCEL
LEC ATM1/0.2:   ATM address     47.00918100000000613E5A2F01.006070174820.02
LEC ATM1/0.2: state ACTIVE event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.3: received RELEASE_COMPLETE
LEC ATM1/0.3:   callid          0x60E8D108
LEC ATM1/0.3:   cause code      0
LEC ATM1/0.3: action A_PROCESS_REL_COMP
LEC ATM1/0.3: action A_TEARDOWN_LEC
LEC ATM1/0.3: sending RELEASE
LEC ATM1/0.3:   callid          0x60EB66D4
LEC ATM1/0.3:   cause code      31
LEC ATM1/0.3: sending RELEASE
LEC ATM1/0.3:   callid          0x60EB7B8C
LEC ATM1/0.3:   cause code      31
LEC ATM1/0.3: sending RELEASE
LEC ATM1/0.3:   callid          0x60EBA3BC
LEC ATM1/0.3:   cause code      31
LEC ATM1/0.3: sending CANCEL
LEC ATM1/0.3:   ATM address     47.00918100000000613E5A2F01.006070174820.03
LEC ATM1/0.3: state ACTIVE event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.2: received RELEASE_COMPLETE
LEC ATM1/0.2:   callid          0x60EB7548
LEC ATM1/0.2:   cause code      0
LEC ATM1/0.2: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.2: state TERMINATING event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.3: received RELEASE_COMPLETE
LEC ATM1/0.3:   callid          0x60EB7B8C
LEC ATM1/0.3:   cause code      0
LEC ATM1/0.3: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.3: state TERMINATING event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.1: received RELEASE_COMPLETE
LEC ATM1/0.1:   callid          0x60EBC458
LEC ATM1/0.1:   cause code      0
LEC ATM1/0.1: action A_PROCESS_REL_COMP
LEC ATM1/0.1: action A_TEARDOWN_LEC
LEC ATM1/0.1: sending RELEASE
LEC ATM1/0.1:   callid          0x60EBD30C
LEC ATM1/0.1:   cause code      31
LEC ATM1/0.1: sending RELEASE
LEC ATM1/0.1:   callid          0x60EBDD28
LEC ATM1/0.1:   cause code      31
LEC ATM1/0.1: sending RELEASE
LEC ATM1/0.1:   callid          0x60EBF174
LEC ATM1/0.1:   cause code      31
LEC ATM1/0.1: sending CANCEL
LEC ATM1/0.1:   ATM address     47.00918100000000613E5A2F01.006070174820.01
LEC ATM1/0.1: state ACTIVE event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.1: received RELEASE_COMPLETE
LEC ATM1/0.1:   callid          0x60EBDD28
LEC ATM1/0.1:   cause code      0
LEC ATM1/0.1: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.1: state TERMINATING event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.2: received RELEASE_COMPLETE
LEC ATM1/0.2:   callid          0x60EB6160
LEC ATM1/0.2:   cause code      0
LEC ATM1/0.2: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.2: state TERMINATING event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.3: received RELEASE_COMPLETE
LEC ATM1/0.3:   callid          0x60EB66D4
LEC ATM1/0.3:   cause code      0
LEC ATM1/0.3: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.3: state TERMINATING event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.2: received RELEASE_COMPLETE
LEC ATM1/0.2:   callid          0x60EB9E48
LEC ATM1/0.2:   cause code      0
```

```
LEC ATM1/0.2: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.2: state TERMINATING event LEC_SIG_RELEASE_COMP => IDLE
LEC ATM1/0.3: received RELEASE_COMPLETE
LEC ATM1/0.3:   callid          0x60EBA3BC
LEC ATM1/0.3:   cause code      0
LEC ATM1/0.3: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.3: state TERMINATING event LEC_SIG_RELEASE_COMP => IDLE
LEC ATM1/0.1: received RELEASE_COMPLETE
LEC ATM1/0.1:   callid          0x60EBD30C
LEC ATM1/0.1:   cause code      0
LEC ATM1/0.1: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.1: state TERMINATING event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.1: received RELEASE_COMPLETE
LEC ATM1/0.1:   callid          0x60EBF174
LEC ATM1/0.1:   cause code      0
LEC ATM1/0.1: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.1: state TERMINATING event LEC_SIG_RELEASE_COMP => IDLE
LEC ATM1/0.2: received CANCEL
LEC ATM1/0.2: state IDLE event LEC_SIG_CANCEL => IDLE
LEC ATM1/0.3: received CANCEL
LEC ATM1/0.3: state IDLE event LEC_SIG_CANCEL => IDLE
LEC ATM1/0.1: received CANCEL
LEC ATM1/0.1: state IDLE event LEC_SIG_CANCEL => IDLE
LEC ATM1/0.1: action A_SHUTDOWN_LEC
LEC ATM1/0.1: sending CANCEL
LEC ATM1/0.1:   ATM address     47.00918100000000613E5A2F01.006070174820.01
LEC ATM1/0.1: state IDLE event LEC_LOCAL_DEACTIVATE => IDLE
LEC ATM1/0.2: action A_SHUTDOWN_LEC
LEC ATM1/0.2: sending CANCEL
LEC ATM1/0.2:   ATM address     47.00918100000000613E5A2F01.006070174820.02
LEC ATM1/0.2: state IDLE event LEC_LOCAL_DEACTIVATE => IDLE
LEC ATM1/0.3: action A_SHUTDOWN_LEC
LEC ATM1/0.3: sending CANCEL
LEC ATM1/0.3:   ATM address     47.00918100000000613E5A2F01.006070174820.03
LEC ATM1/0.3: state IDLE event LEC_LOCAL_DEACTIVATE => IDLE
```

The following output is from the **debug lane client mpoa** command when the **lane** interface is shut down:

```
Router# debug lane client mpoa
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#int atm 1/1/0.1
Router(config-subif)#shutdown
Router(config-subif)#
00:23:32:%LANE-5-UPDOWN:ATM1/1/0.1 elan elan2:LE Client changed state to down
00:23:32:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
00:23:32:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
Router(config-subif)#
Router(config-subif)#
Router(config-subif)#
Router(config-subif)#exit
Router(config)#exit
```

The following output is from the **debug lane client mpoa** command when the **lane** interface is started (not shut down):

```
Router# debug lane client mpoa
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#int atm 1/1/0.1
Router(config-subif)#
Router(config-subif)#
Router(config-subif)#no shutdown
Router(config-subif)#
00:23:39:LEC ATM1/1/0.1:lec_process_lane_tlv:msg LANE_CONFIG_RSP, num_tlvs 14
00:23:39:LEC ATM1/1/0.1:elan id from LECS set to 300
00:23:39:LEC ATM1/1/0.1:lec_process_lane_tlv:msg LANE_JOIN_RSP, num_tlvs 1
00:23:39:LEC ATM1/1/0.1:elan id from LES set to 300
00:23:39:LEC ATM1/1/0.1:lec_append_mpoa_dev_tlv:
00:23:39:LEC ATM1/1/0.1:got mpoa client addr 47.0091810000000050E2097801.0050A
29AF42D.00
00:23:39:%LANE-5-UPDOWN:ATM1/1/0.1 elan elan2:LE Client changed state to up
```

```
00:23:39:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:UP
00:25:57:LEC ATM1/1/0.1:lec_process_lane_tlv:msg LANE_ARP_REQ, num_tlvs 1
00:25:57:LEC ATM1/1/0.1:lec_process_dev_type_tlv:    lec 47.0091810000000050E
2097801.00500B306440.02
    type MPS, mpc 00.000000000000000000000000.000000000000.00
    mps 47.0091810000000050E2097801.00500B306444.00, num_mps_mac 1, mac 0050.0b3
0.6440
00:25:57:LEC ATM1/1/0.1:create mpoa_lec
00:25:57:LEC ATM1/1/0.1:new mpoa_lec 0x617E3118
00:25:57:LEC ATM1/1/0.1:lec_process_dev_type_tlv:type MPS, num       _mps_mac
 1
00:2t 5:57:LEC ATM1/1/0.1:lec_add_mps:
    remote lec 47.0091810000000050E2097801.00500B306440.02
    mps 47.0091810000000050E2097801.00500B306444.00 num_mps_mac 1, mac 0050.0b30
.6440
00:25:57:LEC ATM1/1/0.1:mpoa_device_change:lec_nsap 47.0091810000000050E20978
01.00500B306440.02, appl_type 5
     mpoa_nsap 47.0091810000000050E2097801.00500B306444.00, opcode 4
00:25:57:LEC ATM1/1/0.1:lec_add_mps:add mac 0050.0b30.6440, mps_mac 0x617E372
C
00:25:57:LEC ATM1/1/0.1:mpoa_device_change:lec_nsap 47.0091810000000050E20978
01.00500B306440.02, appl_type 5
     mpoa_nsap 47.0091810000000050E2097801.00500B306444.00, opcode 5
00:25:57:LEC ATM1/1/0.1:     mps_mac 0050.0b30.6440
00:25:57:LEC ATM1/1/0.1:lec_append_mpoa_dev_tlv:
00:25:57:LEC ATM1/1/0.1:got mpoa client addr 47.0091810000000050E2097801.0050A
29AF42D.00
Router(config-subif)#exit
Router(config)#exit
```

The following output is from the **debug lane client mpoa** command when the ATM major interface is shut down:

```
Router# debug lane client mpoa
Router#
conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#int atm 1/1/0
Router(config-if)# shutdown
Router(config-if)#
00:26:28:LANE ATM1/1/0:atm hardware reset
00:26:28:%LANE-5-UPDOWN:ATM1/1/0.1 elan elan2:LE Client changed state to down
00:26:28:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
00:26:28:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
00:26:28:%MPOA-5-UPDOWN:MPC mpc2:state changed to down
00:26:28:LEC ATM1/1/0.1:mpoa_to_lec:appl 6, opcode 0
00:26:30:%LINK-5-CHANGED:Interface ATM1/1/0, changed state to administratively
 down
00:26:30:LANE ATM1/1/0:atm hardware reset
00:26:31:%LINEPROTO-5-UPDOWN:Line protocol on Interface ATM1/1/0, changed stat
e to down
Router(config-if)#
00:26:31:LEC ATM1/1/0.1:mpoa_to_lec:appl 6, opcode 0
00:26:32:LANE ATM1/1/0:atm hardware reset
00:26:32:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
00:26:34:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
Router(config-if)# exit
Router(config)#
exit
```

The following output is from the **debug lane client mpoa** command when the ATM major interface is started:

```
Router# debug lane client mpoa
Router#
conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# int atm 1/1/0
Router(config-if)# no shutdown
00:26:32:LANE ATM1/1/0:atm hardware reset
00:26:32:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
00:26:34:%LINK-3-UPDOWN:Interface ATM1/1/0, changed state to down
00:26:34:LANE ATM1/1/0:atm hardware reset
```

```
00:26:41:%LINK-3-UPDOWN:Interface ATM1/1/0, changed state to up
00:26:42:%LINEPROTO-5-UPDOWN:Line protocol on Interface ATM1/1/0, changed stat
e to up
00:27:10:%LANE-6-INFO:ATM1/1/0:ILMI prefix add event received
00:27:10:LANE ATM1/1/0:prefix add event for 470091810000000050E2097801 ptr=0x6
17BFC0C len=13
00:27:10:    the current first prefix is now:470091810000000050E2097801
00:27:10:%ATMSSCOP-5-SSCOPINIT:- Intf :ATM1/1/0, Event :Rcv End, State :Act
ive.
00:27:10:LEC ATM1/1/0.1:mpoa_to_lec:appl 6, opcode 0
00:27:10:%LANE-3-NOREGILMI:ATM1/1/0.1 LEC cannot register 47.0091810000000050E
2097801.0050A29AF428.01 with ILMI
00:27:10:%LANE-6-INFO:ATM1/1/0:ILMI prefix add event received
00:27:10:LANE ATM1/1/0:prefix add event for 470091810000000050E2097801 ptr=0x6
17B8E6C len=13
00:27:10:    the current first prefix is now:470091810000000050E2097801
00:27:10:%LANE-5-UPDOWN:ATM1/1/0.1 elan elan2:LE Client changed state to down
00:27:10:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
00:27:10:LEC ATM1/1/0.1:mpoa_to_lec:appl 6, opcode 0
00:27:10:%MPOA-5-UPDOWN:MPC mpc2:state changed to up
00:27:10:LEC ATM1/1/0.1:mpoa_to_lec:appl 6, opcode 1
00:27:12:LEC ATM1/1/0.1:lec_process_lane_tlv:msg LANE_CONFIG_RSP, num_tlvs 14
00:27:12:LEC ATM1/1/0.1:elan id from LECS set to 300
00:27:12:LEC ATM1/1/0.1:lec_process_lane_tlv:msg LANE_JOIN_RSP, num_tlvs 1
00:27:12:LEC ATM1/1/0.1:elan id from LES set to 300
00:27:12:LEC ATM1/1/0.1:lec_append_mpoa_dev_tlv:
00:27:12:LEC ATM1/1/0.1:got mpoa client addr 47.0091810000000050E2097801.0050A
29AF42D.00
00:27:12:%LANE-5-UPDOWN:ATM1/1/0.1 elan elan2:LE Client changed state to up
00:27:12:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:UP
Router(config-if)#exit
Router(config)#exit
```

Related Commands

Command	Description
debug modem traffic	Displays MPC debug information.
debug mpoa server	Displays information about the MPOA server.

debug lane config

Note Effective with Cisco IOS Release 15.1M, the **debug lane config** command is not available in Cisco IOS software.

To display information about a LAN Emulation (LANE) configuration server, use the **debug lane config** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug lane config {**all**| **events**| **packets**}

no debug lane config {**all**| **events**| **packets**}

Syntax Description

all	Displays all debugging messages related to the LANE configuration server. The output includes both the **events** and **packets** types of output.
events	Displays only messages related to significant LANE configuration server events.
packets	Displays information on each packet sent or received by the LANE configuration server.

Command Modes

Privileged EXEC

Command History

Release	Modification
15.1M	This command was removed.

Usage Guidelines

The **debug lane config** output is intended to be used primarily by a Cisco technical support representative.

Examples

The following is sample output from the **debug lane config all** command when an interface with LECS, an LES/BUS, and an LEC is shut down:

```
Router# debug lane config all
LECS EVENT ATM1/0: processing interface down transition
LECS EVENT ATM1/0: placed de-register address 0x60E8A824
(47.00918100000000613E5A2F01.006070174823.00) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: placed de-register address 0x60EC4F28
(47.00790000000000000000000.00A03E000001.00) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: placed de-register address 0x60EC5C08
```

```
(47.00918100000000613E5A2F01.006070174823.99) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: tearing down all connexions
LECS EVENT ATM1/0: elan 'xxx' LES 47.00918100000000613E5A2F01.006070174821.01 callId
0x60CE0F58 deliberately being disconnected
LECS EVENT ATM1/0: sending RELEASE for call 0x60CE0F58 cause 31
LECS EVENT ATM1/0: elan 'yyy' LES 47.00918100000000613E5A2F01.006070174821.02 callId
0x60CE2104 deliberately being disconnected
LECS EVENT ATM1/0: sending RELEASE for call 0x60CE2104 cause 31
LECS EVENT ATM1/0: elan 'zzz' LES 47.00918100000000613E5A2F01.006070174821.03 callId
0x60CE2DC8 deliberately being disconnected
LECS EVENT ATM1/0: sending RELEASE for call 0x60CE2DC8 cause 31
LECS EVENT ATM1/0: All calls to/from LECSs are being released
LECS EVENT ATM1/0: placed de-register address 0x60EC4F28
(47.007900000000000000000000.00A03E000001.00) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: ATM_RELEASE_COMPLETE received: callId 0x60CE0F58 cause 0
LECS EVENT ATM1/0: call 0x60CE0F58 cleaned up
LECS EVENT ATM1/0: ATM_RELEASE_COMPLETE received: callId 0x60CE2104 cause 0
LECS EVENT ATM1/0: call 0x60CE2104 cleaned up
LECS EVENT ATM1/0: ATM_RELEASE_COMPLETE received: callId 0x60CE2DC8 cause 0
LECS EVENT ATM1/0: call 0x60CE2DC8 cleaned up
LECS EVENT ATM1/0: UNKNOWN/UNSET: signalling DE-registered
LECS EVENT: UNKNOWN/UNSET: signalling DE-registered
LECS EVENT ATM1/0: UNKNOWN/UNSET: signalling DE-registered
LECS EVENT ATM1/0: placed de-register address 0x60E8A824
(47.00918100000000613E5A2F01.006070174823.00) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: placed de-register address 0x60EC5C08
(47.00918100000000613E5A2F01.006070174823.99) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: tearing down all connexions
LECS EVENT ATM1/0: All calls to/from LECSs are being released
LECS EVENT: config server 56 killed
```

debug lane finder

Note Effective with Cisco IOS Release 15.1M, the **debug lane finder**command is not available in Cisco IOS software.

To display information about the finder internal state machine, use the **debug lane finder** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug lane finder

no debug lane finder

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
15.1M	This command was removed.

Usage Guidelines

The **debug lane finder** command output is intended to be used primarily by a Cisco technical support representative.

Examples

The following is sample output from the **debug lane finder** command when an interface with LECS, LES/BUS, and LEC is shut down:

```
Router# debug lane finder
LECS FINDER ATM1/0.3: user request 1819 of type GET_MASTER_LECS_ADDRESS queued up
LECS FINDER ATM1/0: finder state machine started
LECS FINDER ATM1/0: time to perform a getNext on the ILMI
LECS FINDER ATM1/0: LECS 47.00918100000000613E5A2F01.006070174823.00 deleted
LECS FINDER ATM1/0: ilmi_client_request failed, answering all users
LECS FINDER ATM1/0: answering all requests now
LECS FINDER ATM1/0: responded to user request 1819
LECS FINDER ATM1/0: number of remaining requests still to be processed: 0
LECS FINDER ATM1/0.2: user request 1820 of type GET_MASTER_LECS_ADDRESS queued up
LECS FINDER ATM1/0: finder state machine started
LECS FINDER ATM1/0: time to perform a getNext on the ILMI
LECS FINDER ATM1/0: ilmi_client_request failed, answering all users
LECS FINDER ATM1/0: answering all requests now
LECS FINDER ATM1/0: responded to user request 1820
LECS FINDER ATM1/0: number of remaining requests still to be processed: 0
LECS FINDER ATM1/0.1: user request 1821 of type GET_MASTER_LECS_ADDRESS queued up
LECS FINDER ATM1/0: finder state machine started
LECS FINDER ATM1/0: time to perform a getNext on the ILMI
LECS FINDER ATM1/0: ilmi_client_request failed, answering all users
LECS FINDER ATM1/0: answering all requests now
```

```
LECS FINDER ATM1/0: responded to user request 1821
LECS FINDER ATM1/0: number of remaining requests still to be processed: 0
```

debug lane server

Note Effective with Cisco IOS Release 15.1M, the **debug lane server**command is not available in Cisco IOS software.

To display information about a LAN Emulation (LANE) server, use the **debug lane server** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug lane server [**interface** *interface*]

no debug lane server [**interface** *interface*]

Syntax Description

interface *interface*	(Optional) Limits the debugging output to messages relating to a specific interface or subinterface. If you use this command multiple times with different interfaces, the last interface entered is the one used to filter debugging messages.

Command Modes

Privileged EXEC

Command History

Release	Modification
15.1M	This command was removed.

Usage Guidelines

The **debug lane server** command output is intended to be used primarily by a Cisco technical support representative. The **debug lane server** command can generate a substantial amount of output. Specify a subinterface to decrease the amount of output and focus on the information you need.

Examples

The following is sample output from the **debug lane server** command when an interface with LECS, LES/BUS, and LEC is shut down:

```
Router# debug lane server
LES ATM1/0.1: lsv_lecsAccessSigCB called with callId 0x60CE124C, opcode ATM_RELEASE_COMPLETE
LES ATM1/0.1: disconnected from the master LECS
LES ATM1/0.1: should have been connected, will reconnect in 3 seconds
LES ATM1/0.2: lsv_lecsAccessSigCB called with callId 0x60CE29E0, opcode ATM_RELEASE_COMPLETE
LES ATM1/0.2: disconnected from the master LECS
LES ATM1/0.2: should have been connected, will reconnect in 3 seconds
LES ATM1/0.3: lsv_lecsAccessSigCB called with callId 0x60EB1940, opcode ATM_RELEASE_COMPLETE
LES ATM1/0.3: disconnected from the master LECS
LES ATM1/0.3: should have been connected, will reconnect in 3 seconds
LES ATM1/0.2: elan yyy client 1 lost control distribute
LES ATM1/0.2: elan yyy client 1: lsv_kill_client called
```

```
LES ATM1/0.2: elan yyy client 1 state change Oper -> Term
LES ATM1/0.3: elan zzz client 1 lost control distribute
LES ATM1/0.3: elan zzz client 1: lsv_kill_client called
LES ATM1/0.3: elan zzz client 1 state change Oper -> Term
LES ATM1/0.2: elan yyy client 1 lost MC forward
LES ATM1/0.2: elan yyy client 1: lsv_kill_client called
LES ATM1/0.3: elan zzz client 1 lost MC forward
LES ATM1/0.3: elan zzz client 1: lsv_kill_client called
LES ATM1/0.1: elan xxx client 1 lost control distribute
LES ATM1/0.1: elan xxx client 1: lsv_kill_client called
LES ATM1/0.1: elan xxx client 1 state change Oper -> Term
LES ATM1/0.1: elan xxx client 1 lost MC forward
LES ATM1/0.1: elan xxx client 1: lsv_kill_client called
LES ATM1/0.2: elan yyy client 1 released control direct
LES ATM1/0.2: elan yyy client 1: lsv_kill_client called
LES ATM1/0.3: elan zzz client 1 released control direct
LES ATM1/0.3: elan zzz client 1: lsv_kill_client called
LES ATM1/0.2: elan yyy client 1 MC forward released
LES ATM1/0.2: elan yyy client 1: lsv_kill_client called
LES ATM1/0.2: elan yyy client 1: freeing client structures
LES ATM1/0.2: elan yyy client 1 unregistered 0060.7017.4820
LES ATM1/0.2: elan yyy client 1 destroyed
LES ATM1/0.3: elan zzz client 1 MC forward released
LES ATM1/0.3: elan zzz client 1: lsv_kill_client called
LES ATM1/0.3: elan zzz client 1: freeing client structures
LES ATM1/0.3: elan zzz client 1 unregistered 0060.7017.4820
LES ATM1/0.3: elan zzz client 1 destroyed
LES ATM1/0.1: elan xxx client 1 released control direct
LES ATM1/0.1: elan xxx client 1: lsv_kill_client called
LES ATM1/0.1: elan xxx client 1 MC forward released
LES ATM1/0.1: elan xxx client 1: lsv_kill_client called
LES ATM1/0.1: elan xxx client 1: freeing client structures
LES ATM1/0.1: elan xxx client 1 unregistered 0060.7017.4820
LES ATM1/0.1: elan xxx client 1 destroyed
LES ATM1/0.1: elan xxx major interface state change
LES ATM1/0.1: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.1: shutting down
LES ATM1/0.1: elan xxx: lsv_kill_lesbus called
LES ATM1/0.1: elan xxx: LES/BUS state change operational -> terminating
LES ATM1/0.1: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.2: elan yyy major interface state change
LES ATM1/0.2: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.2: shutting down
LES ATM1/0.2: elan yyy: lsv_kill_lesbus called
LES ATM1/0.2: elan yyy: LES/BUS state change operational -> terminating
LES ATM1/0.2: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.3: elan zzz major interface state change
LES ATM1/0.3: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.3: shutting down
LES ATM1/0.3: elan zzz: lsv_kill_lesbus called
LES ATM1/0.3: elan zzz: LES/BUS state change operational -> terminating
LES ATM1/0.3: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.1: elan xxx: lsv_kill_lesbus called
LES ATM1/0.1: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.1: elan xxx: lsv_kill_lesbus called
LES ATM1/0.1: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.1: elan xxx: stopped listening on addresses
LES ATM1/0.1: elan xxx: all clients killed
LES ATM1/0.1: elan xxx: multicast groups killed
LES ATM1/0.1: elan xxx: addresses de-registered from ilmi
LES ATM1/0.1: elan xxx: LES/BUS state change terminating -> down
LES ATM1/0.1: elan xxx: administratively down
LES ATM1/0.2: elan yyy: lsv_kill_lesbus called
LES ATM1/0.2: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.2: elan yyy: lsv_kill_lesbus called
LES ATM1/0.2: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.2: elan yyy: stopped listening on addresses
LES ATM1/0.2: elan yyy: all clients killed
LES ATM1/0.2: elan yyy: multicast groups killed
LES ATM1/0.2: elan yyy: addresses de-registered from ilmi
LES ATM1/0.2: elan yyy: LES/BUS state change terminating -> down
LES ATM1/0.2: elan yyy: administratively down
LES ATM1/0.3: elan zzz: lsv_kill_lesbus called
```

```
LES ATM1/0.3: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.3: elan zzz: lsv_kill_lesbus called
LES ATM1/0.3: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.3: elan zzz: stopped listening on addresses
LES ATM1/0.3: elan zzz: all clients killed
LES ATM1/0.3: elan zzz: multicast groups killed
LES ATM1/0.3: elan zzz: addresses de-registered from ilmi
LES ATM1/0.3: elan zzz: LES/BUS state change terminating -> down
LES ATM1/0.3: elan zzz: administratively down
LES ATM1/0.3: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.2: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.1: cleanupLecsAccess: discarding all validation requests
```

debug lane signaling

Note Effective with Cisco IOS Release 15.1M, the **debug lane signaling**command is not available in Cisco IOS software.

To display information about LANE Server (LES) and Broadcast and Unknown Server (BUS) switched virtual circuits (SVCs), use the **debug lane signaling** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug lane signaling [**interface** *interface*]

no debug lane signaling [**interface** *interface*]

Syntax Description

interface *interface*	(Optional) Limits the debugging output to messages relating to a specific interface or subinterface. If you use this command multiple times with different interfaces, the last interface entered is the one used to filter debugging messages.

Command Modes

Privileged EXEC

Command History

Release	Modification
15.1M	This command was removed.

Usage Guidelines

The **debug lane signaling** command output is intended to be used primarily by a Cisco technical support representative. The **debug lane signaling** command can generate a substantial amount of output. Specify a subinterface to decrease the amount of output and focus on the information you need.

Examples

The following is sample output from the **debug lane signaling** command when an interface with LECS, LES/BUS, and LEC is shut down:

```
Router# debug lane signaling
LANE SIG ATM1/0.2: received ATM_RELEASE_COMPLETE callid 0x60EB565C cause 0 lv 0x60E8D348
lvstate LANE_VCC_CONNECTED
LANE SIG ATM1/0.2: lane_sig_mc_release: breaking lv 0x60E8D348 from mcg 0x60E97E84
LANE SIG ATM1/0.2: timer for lv 0x60E8D348 stopped
LANE SIG ATM1/0.2: sent ATM_RELEASE request for lv 0x60E8D468 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.2: sent ATM_RELEASE request for lv 0x60E8D3D8 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.2: sent ATM_RELEASE request for lv 0x60E8D2B8 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.3: received ATM_RELEASE_COMPLETE callid 0x60EB5CA0 cause 0 lv 0x60E8BEF4
lvstate LANE_VCC_CONNECTED
LANE SIG ATM1/0.3: lane_sig_mc_release: breaking lv 0x60E8BEF4 from mcg 0x60E9A37C
```

```
LANE SIG ATM1/0.3: timer for lv 0x60E8BEF4 stopped
LANE SIG ATM1/0.3: sent ATM_RELEASE request for lv 0x60E8C014 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.3: sent ATM_RELEASE request for lv 0x60E8BF84 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.3: sent ATM_RELEASE request for lv 0x60E8BE64 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.2: received ATM_RELEASE_COMPLETE callid 0x60EB9040 cause 0 lv 0x60E8D468
lvstate LANE_VCC_DROP_SENT
LANE SIG ATM1/0.2: lane_sig_mc_release: breaking lv 0x60E8D468 from mcg 0x60E97EC8
LANE SIG ATM1/0.2: timer for lv 0x60E8D468 stopped
LANE SIG ATM1/0.3: received ATM_RELEASE_COMPLETE callid 0x60EB97D4 cause 0 lv 0x60E8C014
lvstate LANE_VCC_DROP_SENT
LANE SIG ATM1/0.3: lane_sig_mc_release: breaking lv 0x60E8C014 from mcg 0x60E9A3C0
LANE SIG ATM1/0.3: timer for lv 0x60E8C014 stopped
LANE SIG ATM1/0.1: received ATM_RELEASE_COMPLETE callid 0x60EBCEB8 cause 0 lv 0x60EBBAF0
lvstate LANE_VCC_CONNECTED
LANE SIG ATM1/0.1: lane_sig_mc_release: breaking lv 0x60EBBAF0 from mcg 0x60E8F51C
LANE SIG ATM1/0.1: timer for lv 0x60EBBAF0 stopped
LANE SIG ATM1/0.1: sent ATM_RELEASE request for lv 0x60EBBC10 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.1: sent ATM_RELEASE request for lv 0x60EBBB80 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.1: sent ATM_RELEASE request for lv 0x60EBBA60 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.1: received ATM_RELEASE_COMPLETE callid 0x60EBEB00 cause 0 lv 0x60EBBC10
lvstate LANE_VCC_DROP_SENT
LANE SIG ATM1/0.1: lane_sig_mc_release: breaking lv 0x60EBBC10 from mcg 0x60E8F560
LANE SIG ATM1/0.1: timer for lv 0x60EBBC10 stopped
LANE SIG ATM1/0.2: received ATM_RELEASE_COMPLETE callid 0x60E8B174 cause 0 lv 0x60E8D2B8
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.2: timer for lv 0x60E8D2B8 stopped
LANE SIG ATM1/0.3: received ATM_RELEASE_COMPLETE callid 0x60E8B990 cause 0 lv 0x60E8BE64
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.3: timer for lv 0x60E8BE64 stopped
LANE SIG ATM1/0.2: received ATM_RELEASE_COMPLETE callid 0x60EB7FE0 cause 0 lv 0x60E8D3D8
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.2: timer for lv 0x60E8D3D8 stopped
LANE SIG ATM1/0.3: received ATM_RELEASE_COMPLETE callid 0x60EB8554 cause 0 lv 0x60E8BF84
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.3: timer for lv 0x60E8BF84 stopped
LANE SIG ATM1/0.1: received ATM_RELEASE_COMPLETE callid 0x60EBB6D4 cause 0 lv 0x60EBBA60
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.1: timer for lv 0x60EBBA60 stopped
LANE SIG ATM1/0.1: received ATM_RELEASE_COMPLETE callid 0x60EBE24C cause 0 lv 0x60EBBB80
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.1: timer for lv 0x60EBBB80 stopped
LANE SIG ATM1/0.1: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.1: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.2: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.2: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.3: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.3: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.1: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.1: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.2: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.2: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.3: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.3: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
```

debug lapb

To display all traffic for interfaces using Link Access Procedure, Balanced (LAPB) encapsulation, use the **debug lapb**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug lapb

no debug lapb

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
11.0	This command was introduced prior to this release.

Usage Guidelines

This command displays information on the X.25 Layer 2 protocol. It is useful to users familiar with the LAPB protocol.

You can use the **debug lapb** command to determine why X.25 interfaces or LAPB connections are going up and down. It is also useful for identifying link problems, as evidenced when the **show interfaces** EXEC command displays a high number of rejects or frame errors over the X.25 link.

The **debug lapb** command can generate debugging messages of LAPB on all interfaces configured with the **encapsulation lapb** command or when X.25 traffic is present on interfaces configured with the **encapsulation x25**command. LAPB debugging produces a substantial amount of data and makes debugging very tedious. The problem becomes more severe if the network contains a large number of X.25 interfaces. Therefore the LAPB debugs are set to be available for individual interface.

Caution Because the **debug lapb** command generates a substantial amount of output, use it when the aggregate of all LAPB traffic on X.25 and LAPB interfaces is fewer than five frames per second.

Examples

The following is sample output from the **debug lapb** command (the numbers 1 through 7 at the top of the display have been added in order to aid documentation):

```
1         2   3    4     5    6    7
Serial0: LAPB I CONNECT (5) IFRAME P 2 1
Serial0: LAPB O REJSENT (2) REJ F 3
Serial0: LAPB O REJSENT (5) IFRAME 0 3
Serial0: LAPB I REJSENT (2) REJ (C) 7
Serial0: LAPB I DISCONNECT (2) SABM P
Serial0: LAPB O CONNECT (2) UA F
```

```
Serial0: LAPB O CONNECT (5) IFRAME 0 0
Serial0: LAPB T1 CONNECT 357964 0
```

Each line of output describes a LAPB event. There are two types of LAPB events: frame events (when a frame enters or exits the LAPB) and timer events. In the sample output, the last line describes a timer event; all of the other lines describe frame events. The table below describes the first seven fields.

Table 19: debug lapb Field Descriptions

Field	Description
First field (1)	Interface type and unit number reporting the frame event.
Second field (2)	Protocol providing the information.
Third field (3)	Frame event type. Possible values are as follows: • I--Frame input • O--Frame output • T1--T1 timer expired • T3--Interface outage timer expired • T4--Idle link timer expired
Fourth field (4)	State of the protocol when the frame event occurred. Possible values are as follows: • BUSY (RNR frame received) • CONNECT • DISCONNECT • DISCSENT (disconnect sent) • ERROR (FRMR frame sent) • REJSENT (reject frame sent) • SABMSENT (SABM frame sent)
Fifth field (5)	In a frame event, this value is the size of the frame (in bytes). In a timer event, this value is the current timer value (in milliseconds).

Field	Description
Sixth field (6)	In a frame event, this value is the frame type name. Possible values for f rame type names are as follows: • DISC--Disconnect • DM--Disconnect mode • FRMR--Frame reject • IFRAME--Information frame • ILLEGAL--Illegal LAPB frame • REJ--Reject • RNR--Receiver not ready • RR--Receiver ready • SABM--Set asynchronous balanced mode • SABME--Set asynchronous balanced mode, extended • UA--Unnumbered acknowledgment In a T1 timer event, this value is the number of retransmissions already attempted.
Seventh field (7) (This field will not print if the frame control field is required to appear as either a command or a response, and that frame type is correct.)	This field is present only in frame events. It describes the frame type identified by the LAPB address and Poll/Final bit. Possible values are as follows: • (C)--Command frame • (R)--Response frame • P--Command/Poll frame • F--Response/Final frame • /ERR--Command/Response type is invalid for the control field. An ?ERR generally means that the data terminal equipment (DTE)/data communications equipment (DCE) assignments are not correct for this link. • BAD-ADDR--Address field is neither Command nor Response

A timer event displays only the first six fields of **debug lapb** command output. For frame events, however, the seventh field documents the LAPB control information present in the frame. Depending on the value of the frame type name shown in the sixth field, the seventh field may or may not appear.

After the Poll/Final indicator, depending on the frame type, three different types of LAPB control information can be printed.

For information frames, the value of the N(S) field and the N(R) field will be printed. The N(S) field of an information frame is the sequence number of that frame, so this field will rotate between 0 and 7 for (modulo 8 operation) or 0 and 127 (for modulo 128 operation) for successive outgoing information frames and (under normal circumstances) also will rotate for incoming information frame streams. The N(R) field is a "piggybacked" acknowledgment for the incoming information frame stream; it informs the other end of the link which sequence number is expected next.

RR, RNR, and REJ frames have an N(R) field, so the value of that field is printed. This field has exactly the same significance that it does in an information frame.

For the FRMR frame, the error information is decoded to display the rejected control field, V(R) and V(S) values, the Response/Command flag, and the error flags WXYZ.

In the following example, the output shows an idle link timer action (T4) where the timer expires twice on an idle link, with the value of T4 set to five seconds:

```
Serial2: LAPB T4 CONNECT 255748
Serial2: LAPB O CONNECT (2) RR P 5
Serial2: LAPB I CONNECT (2) RR F 5
Serial2: LAPB T4 CONNECT 260748
Serial2: LAPB O CONNECT (2) RR P 5
Serial2: LAPB I CONNECT (2) RR F 5
```

The next example shows an interface outage timer expiration (T3):

```
Serial2: LAPB T3 DISCONNECT 273284
```

The following example output shows an error condition when no DCE to DTE connection exists. Note that if a frame has only one valid type (for example, a SABM can only be a command frame), a received frame that has the wrong frame type will be flagged as a receive error (R/ERR in the following output). This feature makes misconfigured links (DTE-DTE or DCE-DCE) easy to spot. Other less common errors will also be highlighted, such as a too-short or too-long frame or an invalid address (neither command nor response).

```
Serial2: LAPB T1 SABMSENT 1026508 1
Serial2: LAPB O SABMSENT (2) SABM P
Serial2: LAPB I SABMSENT (2) SABM (R/ERR)
Serial2: LAPB T1 SABMSENT 1029508 2
Serial2: LAPB O SABMSENT (2) SABM P
Serial2: LAPB I SABMSENT (2) SABM (R/ERR)
```

The output in the next example shows tha the router is misconfigured and has a standard (modulo 8) interface connected to an extended (modulo 128) interface. This condition is indicated by the SABM balanced mode and SABME balanced mode extended messages appearing on the same interface.

```
Serial2: LAPB T1 SABMSENT 1428720 0
Serial2: LAPB O SABMSENT (2) SABME P
Serial2: LAPB I SABMSENT (2) SABM P
Serial2: LAPB T1 SABMSENT 1431720 1
Serial2: LAPB O SABMSENT (2) SABME P
Serial2: LAPB I SABMSENT (2) SABM P
```

The output in the next example shows that the **debug lapb**command is set for a single interface; that is, interface 0/0.

```
Serial0/0: LAPB O CONNECT (17) IFRAME 1 7
Serial0/0: LAPB I CONNECT (5) IFRAME 7 2
Serial0/0: LAPB I CONNECT (6) IFRAME 0 2
Serial0/0: LAPB O CONNECT (2) RR (R) 1
Serial0/0: LAPB O CONNECT (50) IFRAME 2 1
Serial0/0: LAPB I CONNECT (15) IFRAME 1 2
Serial0/0: LAPB O CONNECT (5) IFRAME 3 2
```

debug lapb-ta

To display debugging messages for Link Access Procedure, Balanced-Terminal Adapter (LAPB-TA), use the **debug lapb-ta** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug lapb-ta [**error**| **event**| **traffic**]

no debug lapb-ta [**error**| **event**| **traffic**]

Syntax Description

error	(Optional) Displays LAPB-TA errors.
event	(Optional) Displays LAPB-TA normal events.
traffic	(Optional) Displays LAPB-TA in/out traffic data.

Command Default

Debugging for LAPB-TA is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(4)T	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Examples

The following is sample output from the **debug lapb-ta** command with the **error**, **event**, and **traffic** keywords activated:

```
Router# debug lapb-ta error
LAPB-TA error debugging is on
Router# debug lapb-ta event
LAPB-TA event debugging is on
Router# debug lapb-ta traffic
LAPB-TA traffic debugging is on
Mar  9 12:11:36.464:LAPB-TA:Autodetect trying to detect LAPB on
BR3/0:1
Mar  9 12:11:36.464:  sampled pkt: 2 bytes: 1 3F.. match
Mar  9 12:11:36.468:LAPBTA:get_ll_config:BRI3/0:1
Mar  9 12:11:36.468:LAPBTA:line 130 allocated for BR3/0:1
Mar  9 12:11:36.468:LAPBTA:process 79
Mar  9 12:11:36.468:BR3/0:1:LAPB-TA started
Mar  9 12:11:36.468:LAPBTA:service change:LAPB physical layer up,
context 6183E144 interface up, protocol down
Mar  9 12:11:36.468:LAPBTA:service change:, context 6183E144 up
Mar  9 12:11:36.468:LAPB-TA:BR3/0:1, 44 sent
2d14h:%LINEPROTO-5-UPDOWN:Line protocol on Interface BRI3/0:1, changed state to up
```

```
2d14h:%ISDN-6-CONNECT:Interface BRI3/0:1 is now connected to 60213
Mar  9 12:11:44.508:LAPB-TA:BR3/0:1, 1 rcvd
Mar  9 12:11:44.508:LAPB-TA:BR3/0:1, 3 sent
Mar  9 12:11:44.700:LAPB-TA:BR3/0:1, 1 rcvd
Mar  9 12:11:44.700:LAPB-TA:BR3/0:1, 3 sent
Mar  9 12:11:44.840:LAPB-TA:BR3/0:1, 1 rcvd
Mar  9 12:11:44.840:LAPB-TA:BR3/0:1, 14 sent
Mar  9 12:11:45.852:LAPB-TA:BR3/0:1, 1 rcvd
Mar  9 12:11:46.160:LAPB-TA:BR3/0:1, 2 rcvd
Mar  9 12:11:47.016:LAPB-TA:BR3/0:1, 1 rcvd
Mar  9 12:11:47.016:LAPB-TA:BR3/0:1, 10 sent
```

debug lat packet

To display information on all local-area transport (LAT) events, use the **debug lat packet** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug lat packet

no debug lat packet

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

For each datagram (packet) received or sent, a message is logged to the console.

Caution This command severely impacts LAT performance and is intended for troubleshooting use only.

Examples

The following is sample output from the **debug lat packet** command:

```
Router# debug lat packet
LAT: I int=Ethernet0, src=0000.0c01.0509, dst=0900.2b00.000f, type=0, M=0, R=0
LAT: I int=Ethernet0, src=0800.2b11.2d13, dst=0000.0c01.7876, type=A, M=0, R=0
LAT: O dst=0800.2b11.2d13, int=Ethernet0, type= A, M=0, R=0, len= 20, next 0 ref 1
```

The second line of output describes a packet that is input to the router. The table below describes the fields in this line.

Table 20: debug lat packet Field Descriptions

Field	Description
LAT:	Indicates that this display shows LAT debugging output.
I	Indicates that this line of output describes a packet that is input to the router (I) or output from the router (O).
int = Ethernet0	Indicates the interface on which the packet event took place.
src = 0800.2b11.2d13	Indicates the source address of the packet.
dst=0000.0c01.7876	Indicates the destination address of the packet.

Field	Description
type=A	Indicates the message type (in hexadecimal notation). Possible values are as follows: • 0 = Run Circuit • 1 = Start Circuit • 2 = Stop Circuit • A = Service Announcement • C = Command • D = Status • E = Solicit Information • F = Response Information

The third line of output describes a packet that is output from the router. The table below describes the last three fields in this line.

Table 21: debug lat packet Field Descriptions

Field	Description
len= 20	Indicates the length (in hexadecimal notation) of the packet (in bytes).
next 0	Indicates the link on the transmit queue.
ref 1	Indicates the count of packet users.

debug ldap

To enable debugging for Lightweight Directory Access Protocol (LDAP) configuration, use the **debug ldap** command in privileged EXEC mode. To disable debugging, use the no form of this command.

debug ldap {**all**| **error**| **event**| **legacy**| **packet**}

no debug ldap {**all**| **error**| **event**| **legacy**| **packet**}

Syntax Description

all	Displays all event, legacy, and packet related messages.
error	Displays error messages about the local authentication server.
event	Displays debug messages related to LDAP proxy events.
legacy	Displays legacy messages.
packet	Displays the content of the RADIUS packets that are sent and received.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.1(1)T	This command was introduced.

Examples

The following is sample output from the **debug ldap legacy** command:

```
Router# debug ldap legacy
put_filter "(&(objectclass=*)(cn=firewall_user))"
put_filter: AND
put_filter_list "(objectclass=*)(cn=firewall_user)"
put_filter "(objectclass=*)"
put_filter: simple
put_filter "(cn=firewall_user)"
put_filter: simple
Doing socket writeldap_result
wait4msg (timeout 0 sec, 1 usec)
ldap_select_fd_wait (select)
ldap_read_activity lc 0x6804D354
Doing socket read
LDAP-TCP:Bytes read = 1478
ldap_match_request succeeded for msgid 2 h 0
ldap_get_dn
```

```
ldap_get_dn
ldap_msgfree
ldap_result
wait4msg (timeout 0 sec, 1 usec)
ldap_read_activity lc 0x6804D354
ldap_match_request succeeded for msgid 2 h 0
changing lr 0x6774F8D4 to COMPLETE as no continuations
removing request 0x6774F8D4 from list as lm 0x681C9B78 all 0
ldap_msgfree
ldap_msgfree
ldap_parse_result
ldap_parse_result
ldap_req_encode
Doing socket writeldap_msgfree
ldap_result
wait4msg (timeout 0 sec, 1 usec)
ldap_select_fd_wait (select)
ldap_result
wait4msg (timeout 0 sec, 1 usec)
ldap_select_fd_wait (select)
ldap_read_activity lc 0x6804D354
Doing socket read
LDAP-TCP:Bytes read = 22
ldap_match_request succeeded for msgid 3 h 0
changing lr 0x6774F8D4 to COMPLETE as no continuations
removing request 0x6774F8D4 from list as lm 0x681C9B78 all 0
ldap_msgfree
ldap_msgfree
ldap_parse_result
ldap_parse_result
ldap_msgfree
ldap_result
wait4msg (timeout 0 sec, 1 usec)
ldap_select_fd_wait (select)
```

Related Commands

Command	Description
ipv4 (ldap)	Creates an IPv4 address within an LDAP server address pool
ldap server	Defines an LDAP server and enters LDAP server configuration mode.
transport port (ldap)	Configures the transport protocol for establishing a connection with the LDAP server.

debug lex rcmd

To debug LAN Extender remote commands, use the **debug lex rcmd** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug lex rcmd

no debug lex rcmd

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Examples

The following is sample output from the **debug lex rcmd** command:

```
Router# debug lex rcmd
LEX-RCMD: "shutdown" command received on unbound serial interface- Serial0
LEX-RCMD: Lex0: "inventory" command received
Rcvd rcmd: FF 03 80 41 41 13 00 1A 8A 00 00 16 01 FF 00 00
Rcvd rcmd: 00 02 00 00 07 5B CD 15 00 00 0C 01 15 26
LEX-RCMD: ACK or response received on Serial0 without a corresponding ID
LEX-RCMD: REJ received
LEX-RCMD: illegal CODE field received in header: <number>
LEX-RCMD: illegal length for Lex0: "lex input-type-list"
LEX-RCMD: Lex0 is not bound to a serial interface
LEX-RCMD: encapsulation failure
LEX-RCMD: timeout for Lex0: "lex priority-group" command
LEX-RCMD: re-transmitting Lex0: "lex priority-group" command
LEX-RCMD: lex_setup_and_send called with invalid parameter
LEX-RCMD: bind occurred on shutdown LEX interface
LEX-RCMD: Serial0- No free Lex interface found with negotiated MAC address 0000.0c00.d8db
LEX-RCMD: No active Lex interface found for unbind
```

The following output indicates that a LAN Extender remote command packet was received on a serial interface that is not bound to a LAN Extender interface:

```
LEX-RCMD: "shutdown" command received on unbound serial interface- Serial0
```

This message can occur for any of the LAN Extender remote commands. Possible causes of this message are as follows:

- FLEX state machine software error
- Serial line momentarily goes down, which is detected by the host but not by FLEX

The following output indicates that a LAN Extender remote command response has been received. The hexadecimal values are for internal use only.

```
LEX-RCMD: Lex0: "inventory" command received
Rcvd rcmd: FF 03 80 41 41 13 00 1A 8A 00 00 16 01 FF 00 00
Rcvd rcmd: 00 02 00 00 07 5B CD 15 00 00 0C 01 15 26
```

The following output indicates that when the host router originates a LAN Extender remote command to FLEX, it generates an 8-bit identifier that is used to associate a command with its corresponding response:

```
LEX-RCMD: ACK or response received on Serial0 without a corresponding ID
```

This message could be displayed for any of the following reasons:

- FLEX was busy at the time that the command arrived and could not send an immediate response. The command timed out on the host router and then FLEX finally sent the response.
- Transmission error.
- Software error.

Possible responses to Config-Request are Config-ACK, Config-NAK, and Config-Rej. The following output shows that some of the options in the Config-Request are not recognizable or are not acceptable to FLEX due to transmission errors or software errors:

```
LEX-RCMD: REJ received
```

The following output shows that a LAN Extender remote command response was received but that the CODE field in the header was incorrect:

```
LEX-RCMD: illegal CODE field received in header: <number>
```

The following output indicates that a LAN Extender remote command response was received but that it had an incorrect length field. This message can occur for any of the LAN Extender remote commands.

```
LEX-RCMD: illegal length for Lex0: "lex input-type-list"
```

The following output shows that a host router was about to send a remote command when the serial link went down:

```
LEX-RCMD: Lex0 is not bound to a serial interface
```

The following output shows that the serial encapsulation routine of the interface failed to encapsulate the remote command datagram because the LEX-NCP was not in the OPEN state. Due to the way the PPP state machine is implemented, it is normal to see a single encapsulation failure for each remote command that gets sent at bind time.

```
LEX-RCMD: encapsulation failure
```

The following output shows that the timer expired for the given remote command without having received a response from the FLEX device. This message can occur for any of the LAN Extender remote commands.

```
 LEX-RCMD: timeout for Lex0: "lex priority-group" command
```

This message could be displayed for any of the following reasons:

- FLEX too busy to respond
- Transmission failure
- Software error

The following output indicates that the host is resending the remote command after a timeout:

```
LEX-RCMD: re-transmitting Lex0: “lex priority-group” command
```

The following output indicates that an illegal parameter was passed to the lex_setup_and_send routine. This message could be displayed due to a host software error.

```
LEX-RCMD: lex_setup_and_send called with invalid parameter
```

The following output is informational and shows when a bind occurs on a shutdown interface:

```
LEX-RCMD: bind occurred on shutdown LEX interface
```

The following output shows that the LEX-NCP reached the open state and a bind operation was attempted with the FLEX's MAC address, but no free LAN Extender interfaces were found that were configured with

that MAC address. This output can occur when the network administrator does not configure a LAN Extender interface with the correct MAC address.

```
LEX-RCMD: Serial0- No free Lex interface found with negotiated MAC address 0000.0c00.d8db
```

The following output shows that the serial line that was bound to the LAN Extender interface went down and the unbind routine was called, but when the list of active LAN Extender interfaces was searched, the LAN Extender interface corresponding to the serial interface was not found. This output usually occurs because of a host software error.

```
LEX-RCMD: No active Lex interface found for unbind
```

debug license

To enable controlled Cisco IOS software license debugging activity on a device, use the **debug license** command in privileged EXEC mode. To disable debugging, use the no form of this command.

debug license {**agent** {**all**| **error**}| **core** {**all**| **errors**| **events**}| **errors**| **events**| **ipc**}

no debug license {**agent** {**all**| **error**}| **core** {**all**| **errors**| **events**}| **errors**| **events**| **ipc**}

Cisco ASR 1001 Router Platforms

debug license {**core** {**all**| **errors**| **events**}| **errors**| **ipc**}

no debug license {**core** {**all**| **errors**| **events**}| **errors**| **ipc**}

Syntax Description

agent	Debugs license agent information. • **all** --Debugs all license agent messages. • **error** --Debugs only license agent error messages.
core	Debugs messages from a license core module. • **all** --Debugs all license core messages • **errors** --Debugs only license core error messages • **events** --Debugs only license core event messages.
errors	Debugs license warnings and errors.
events	Debugs license event messages.
ipc	Debugs license interprocess communication (IPC) messages.

Command Default

Debugging is disabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(35)SE2	This command was introduced.

Release	Modification
12.4(15)XZ	This command was integrated into Cisco IOS Release 12.4(15)XZ.
12.4(20)T	This command was integrated into Cisco IOS Release 12.4(20)T.
Cisco IOS XE Release 3.2S	This command was implemented on the Cisco ASR 1001 router.

Usage Guidelines

Use this command to help troubleshoot issues with licenses on a device.

On the Cisco ASR 1001 router, the output from the **debug license** command is not in standard IOS format. You must execute the **request platform software trace rotate all** privileged EXEC command to make the output in the log files in the bootflash:tracelogs directory.

Examples

The following example shows how to enable debugging for license warnings and errors on a router:

```
Router# debug license errors
```

The following example shows how to enable debugging for all license agent information on a switch:

```
Switch# debug license agent all
license agent app https[0x43FBC7C]: urlhook function
license agent app https[0x43FBC7C]: https action function
LIC_AGENT:Processing XML message
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
<SOAP:Header>
<clm:Header version="1.0" xmlns:clm="http://www.cisco.com/clm">
<clm:Time>2003-04-23T20:27:19.827Z</clm:Time>
</clm:Header>
</SOAP:Header>
<SOAP:Body>
<lica:request xmlns:lica="http://www.cisco.com/clm">
<lica:installRequest>
<lica:license encoding="BASE64">
PENJU0NPX1dUX0FSVElGQUNUUyB2ZXJzaW9uPSIxLjAiPjxDSVNDT19XVF9MSUNFTlNFIHZlcnNp
b249IjEuMCI+PEZFQVRVUkVfTkFNRT5pcGJhc2U8L0ZFQVRVUkVfTkFNRT48RkVBVFVSRV9WRVJT
SU9OPjEuMDwvRkVBVFVSRV9WRVJTSU9OPjxVREk+PFBJRD5CVUxMU0VZRTI0PC9QSUQ+PFNOPkNB
VDEwMDZSMEU4PC9TTj48L1VEST48U09VUkNFPkNpc2NvIEhRPC9TT1VSQ0U+PENSRUFURV9EQVRF
PjIwMDYtMTEtMjJUMDA6MzM6NTA8L0NSRUFURV9EQVRFPjxMSUNFTlNFX0xJTkVfSEFTCBoYXNo
QWxnbz0iU0hBMSI+NDJiNFVWWFpOd3pJK0ZNdEV6QlNZSDRWdzFFPTwvTElDRU5TRV9MSU5FX0hB
U0g+PFRZUEU+UkVHVUxBUjwvVFlQRT48TElDRU5TRV9MSU5FPjwhW0NEQVRBWzExIGlwYmFzZSAx
LjAgTE9ORyBOT1JNQUwgU1RBTkRBTE9ORSBFWENMIElORklOVSRFX0tFWVMgSU5GSU5JVEVfS0VZ
UyBORVZFUiBORVZFUiBOaUwgU0xNX0NPREUgQ0xfTkRfTENLIE5pTCAqMVZBUlY5WlJESzREOU5U
NDAwIE5pTCBOaUwgTmlMIDVfTUlOUyA8VURJPjxQSUQ+QlVMTFNFWUUyNDwvUElEPjxTTj5DQVQx
MDA2UjBFODwvU04+PC9VREk+IGUxWW8wS1U2VnJLOnBJZXRib1dJVkEyZlVaVGdieUlEaklHWERR
VXc3dkxOYWlXRzZ0dUJOMG5lTXpKaHpzQ2tMN1l3TWFxS2paem05YW5FbVJHUUVPTHlDdmRVZksw
QmNLN0pPcnZsUkw0VjMyJDxXTEM+QVFFQklRQUIvLy9GbS8vWDkybThNb0NOZkVMSHJiVzRjWDFM
ZGNpdDNMVU5GWlV1OWppT0phcXB5Q2N6TTFpaUlKbVE3NEd5WHJFY3F2UGlBbVdTYUVtVVQ1NnJz
dGs2Z3ZtaitFUUtSZkQ5QTBpbWUxY3pyZEt4ZklMVDBMYVhUNDE2bndtZnA5MlR5YTZ2SVE0Rm5s
QmRxSjFzTXpYZVNxOFBtVmNUVTlBNG85aGlsOXZLdXI4TjlGODg1RDlHVkYwYkpIY2lUNU09PC9X
TEM+XV0+PC9MSUNFTlNFX0xJTkU+PFVTRVJfTU9ESUZJQUJMRV9DT01NRU5UIGZpZWxkUmVzdHJp
Y3Rpb25zPSJNYXggOTkgQVNDSUkgY2hhcmFjdGVycyBpbiBsZW5ndGguIj48L1VTRVJfTU9ESUZJ
QUJMRV9DT01NRU5UPjwvQ0lTQ09fV1RfTElDRU5TRT48L0NJU0NPX1dUX0FSVElGQUNUUz4=
</lica:license>
</lica:installRequest>
</lica:request>
</SOAP:Body>
</SOAP:Envelope>
LIC_AGENT: XML received opcode(1)
LIC_AGENT: License ipbase
%IOS_LICENSE_IMAGE_APPLICATION-6-LICENSE_LEVEL: Next reboot level = ipbase and License =
```

```
ipbase
LIC_AGENT: Notification Event type = 1 License Installed
LIC_AGENT: Notification Event type = 13 License Annotate
```

debug link monitor

To display the statistics of the executing process, use the **debug link monitor** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug link monitor

no debug link monitor

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(1)	This command was introduced.

Usage Guidelines

This command is used to display the statistics, which are used for debugging the status of the various conditions occurred during execution of the monitoring process.

Examples

The following example enables link monitoring statistics:

```
Router# debug link monitor
%DEBUG-ENABLED Error Rate Link Monitor
```

The following example disables link monitoring statistics:

```
Router# no debug link monitor

%DEBUG-DISABLED Error Rate Link Monitor
```

Related Commands

Command	Description
debug all	Enables debugging for link monitoring.
no debug all	Disables debugging for link monitoring.
clear counters	Clears show interface counters on all interfaces.
show link monitor debug	Show link monitor error statistics.

debug list

To filter debugging information on a per-interface or per-access list basis, use the **debug list** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug list [*list*] [*interface*]

no debug list [*list*] [*interface*]

Syntax Description

list	(Optional) An access list number in the range from 1100 to 1199.
interface	(Optional) The interface type. Allowed values are the following: • **channel** --IBM Channel interface • **ethernet** --IEEE 802.3 • **fddi** --ANSI X3T9.5 • **null** --Null interface • **serial** --Serial • **tokenring** --IEEE 802.5 • **tunnel** --Tunnel interface

Command Modes

Privileged EXEC

Usage Guidelines

The **debug list** command is used with other **debug** commands for specific protocols and interfaces to filter the amount of debug information that is displayed. In particular, this command is designed to filter specific physical unit (PU) output from bridging protocols. The **debug list** command is supported with the following commands:

- **debug arp**
- **debug llc2 errors**
- **debug llc2 packets**
- **debug llc2 state**
- **debug rif**
- **debug sdlc**
- **debug token ring**

Note All **debug** commands that support access list filtering use access lists in the range from 1100 to 1199. The access list numbers shown in the examples are merely samples of valid numbers.

Examples

To use the **debug list** command on only the first of several Logical Link Control, type 2 (LLC2) connections, use the **show llc2** command to display the active connections:

```
Router# show llc2
SdllcVirtualRing2008 DTE: 4000.2222.22c7 4000.1111.111c 04 04 state NORMAL
SdllcVirtualRing2008 DTE: 4000.2222.22c8 4000.1111.1120 04 04 state NORMAL
SdllcVirtualRing2008 DTE: 4000.2222.22c1 4000.1111.1104 04 04 state NORMAL
```

Next, configure an extended bridging access list, numbered 1103, for the connection you want to filter:

```
access-list 1103 permit 4000.1111.111c 0000.0000.0000 4000.2222.22c7 0000.0000.0000 0xC 2
eq 0x404
```

The convention for the LLC **debug list** command filtering is to use dmac = 6 bytes, smac = 6 bytes, dsap_offset = 12, and ssap_offset = 13.

Finally, you invoke the following **debug** commands:

```
Router# debug list 1103
Router# debug llc2 packet
LLC2 Packets debugging is on
for access list: 1103
```

To use the **debug list** command for Synchronous Data Link Control (SDLC) connections, with the exception of address 04, create access list 1102 to deny the specific address and permit all others:

```
access-list 1102 deny 0000.0000.0000 0000.0000.0000 0000.0000.0000 0000.0000.0000 0xC 1 eq
 0x4
access-list 1102 permit 0000.0000.0000 0000.0000.0000 0000.0000.0000 0000.0000.0000
```

The convention is to use dmac = 0.0.0, smac = 0.0.0, and sdlc_frame_offset = 12.

Invoke the following **debug** commands:

```
Router# debug list 1102
Router# debug sdlc
SDLC link debugging is on
for access list: 1102
```

To enable SDLC debugging (or debugging for any of the other supported protocols) for a specific interface rather than for all interfaces on a router, use the following commands:

```
Router# debug list serial 0
Router# debug sdlc
SDLC link debugging is on
for interface: Serial0
```

To enable Token Ring debugging between two MAC address, 0000.3018.4acd and 0000.30e0.8250, configure an extended bridging access list 1106:

```
access-list 1106 permit 0000.3018.4acd 8000.0000.0000 0000.30e0.8250 8000.0000.0000
access-list 1106 permit 0000.30e0.8250 8000.0000.0000 0000.3018.4acd 8000.0000.0000
```

Invoke the following **debug** commands:

```
Router# debug list 1106
Router# debug token ring
Token Ring Interface debugging is on
for access list: 1106
```

To enable routing information field (RIF) debugging for a single MAC address, configure an access list 1109:

```
access-list 1109 permit permit 0000.0000.0000 ffff.ffff.ffff 4000.2222.22c6 0000.0000.0000
```

Invoke the following **debug** commands:

```
Router# debug list 1109
Router# debug rif
RIF update debugging is on
for access list: 1109
```

Related Commands

Command	Description
debug llc2 errors	Displays LLC2 protocol error conditions or unexpected input.
debug llc2 packet	Displays all input and output from the LLC2 protocol stack.
debug llc2 state	Displays state transitions of the LLC2 protocol.
debug rif	Displays information on entries entering and leaving the RIF cache.
debug rtsp	Displays information on SDLC frames received and sent by any router serial interface involved in supporting SDLC end station functions.
debug token ring	Displays messages about Token Ring interface activity.

debug llc2 dynwind

To display changes to the dynamic window over Frame Relay, use the **debug llc2 dynwind**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug llc2 dynwind

no debug llc2 dynwind

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Examples

The following is sample output from the **debug llc2 dynwind** command:

```
Router# debug llc2 dynwind
LLC2/DW: BECN received! event REC_I_CMD, Window size reduced to 4
LLC2/DW: 1 consecutive I-frame(s) received without BECN
LLC2/DW: 2 consecutive I-frame(s) received without BECN
LLC2/DW: 3 consecutive I-frame(s) received without BECN
LLC2/DW: 4 consecutive I-frame(s) received without BECN
LLC2/DW: 5 consecutive I-frame(s) received without BECN
LLC2/DW: Current working window size is 5
```

In this example, the router receives a backward explicit congestion notification (BECN) and reduces the window size to 4. After receiving five consecutive I frames without a BECN, the router increases the window size to 5.

Related Commands

Command	Description
debug llc2 errors	Displays LLC2 protocol error conditions or unexpected input.
debug llc2 packet	Displays all input and output from the LLC2 protocol stack.
debug llc2 state	Displays state transitions of the LLC2 protocol.

debug llc2 errors

To display Logical Link Control, type 2 (LLC2) protocol error conditions or unexpected input, use the **debug llc2 errors**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug llc2 errors

no debug llc2 errors

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Examples

The following is sample output from the **debug llc2 errors** command from a router ignoring an incorrectly configured device:

```
Router# debug llc2 errors
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
```

Each line of output contains the remote MAC address, the local MAC address, the remote service access point (SAP), and the local SAP. In this example, the router receives unsolicited RR frames marked as responses.

Related Commands

Command	Description
debug list	Filters debugging information on a per-interface or per-access list basis.
debug llc2 dynwind	Displays changes to the dynamic window over Frame Relay.
debug llc2 packet	Displays all input and output from the LLC2 protocol stack.
debug llc2 state	Displays state transitions of the LLC2 protocol.

debug llc2 packet

To display all input and output from the Logical Link Control, type 2 (LLC2) protocol stack, use the **debug llc2 packet** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug llc2 packet

no debug llc2 packet

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

This command also displays information about some error conditions as well as internal interactions between the Common Link Services (CLS) layer and the LLC2 layer.

Examples

The following is sample output from the **debug llc2 packet** command from the router sending ping data back and forth to another router:

```
Router# debug llc2 packet
LLC: llc2_input
401E54F0:                            10400000             .@..
401E5500: 303A90CF 0006F4E1 2A200404 012B5E    0:.O..ta* ...+
LLC: i REC_RR_CMD N(R)=21 p/f=1
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 NORMAL REC_RR_CMD (3)
LLC (rs): 0006.f4e1.2a20 0000.303a.90cf 04 04 REC_RR_CMD N(R)=42
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 txmt RR_RSP N(R)=20 p/f=1
LLC: llc_sendframe
401E5610:              0040 0006F4E1 2A200000        .@..ta* ..
401E5620: 303A90CF 04050129 00               N  0:.O...).       2012
LLC: llc_sendframe
4022E3A0:                        0040 0006F4E1            .@..ta
4022E3B0: 2A200000 303A90CF 04042A28 2C000202  * ..0:.O..*(,...
4022E3C0: 00050B90 A02E0502 FF0003D1 004006C1  .... ......Q.@.A
4022E3D0: D7C9D5C    0.128
 C400130A C1D7D7D5 4BD5F2F0  WIUGD...AWWUKUrp
4022E3E0: F1F30000 011A6071 00010860 D7027000  qs....`q...`W.p.
4022E3F0: 00003B00 1112FF01 03000243 6973636F  ..;........Cisco
4022E400: 20494F53 69                           IOSi
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 txmt I N(S)=21 N(R)=20 p/f=0 size=90
LLC: llc2_input
401E5620:                  10400000 303A90CF          .@..0:.O
401E5630: 0006F4E1 2A200404 282C2C00 02020004  ..ta* ..(,,.....
401E5640: 03902000 1112FF01 03000243 6973636F  .. ........Cisco
401E5650: 20494F53 A0                           IOS
LLC: i REC_I_CMD N(R)=22 N(S)=20 V(R)=20 p/f=0
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 NORMAL REC_I_CMD (1)
LLC (rs): 0006.f4e1.2a20 0000.303a.90cf 04 04 REC_I_CMD N(S)=20 V(R)=20
LLC (rs): 0006.f4e1.2a20 0000.303a.90cf 04 04 REC_I_CMD N(R)=44
LLC: INFO: 0006.f4e1.2a20 0000.303a.90cf 04 04 v(r) 20
```

The first three lines indicate that the router has received some input from the link:

```
LLC: llc2_input
401E54F0:                            10400000             .@..
401E5500: 303A90CF 0006F4E1 2A200404 012B5E    0:.O..ta* ...+
```

The next line indicates that this input was an RR command with the poll bit set. The other router has received sequence number 21 and is waiting for the final bit.

```
LLC: i REC_RR_CMD N(R)=21 p/f=1
```

The next two lines contain the MAC addresses of the sender and receiver, and the state of the router when it received this frame:

```
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 NORMAL REC_RR_CMD (3)
LLC (rs): 0006.f4e1.2a20 0000.303a.90cf 04 04 REC_RR_CMD N(R)=42
```

The next four lines indicate that the router is sending a response with the final bit set:

```
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 txmt RR_RSP N(R)=20 p/f=1
LLC: llc_sendframe
401E5610:              0040 0006F4E1 2A200000        .@..ta* ..
401E5620: 303A90CF 04050129 00               N  0:.O...).       2012
```

Related Commands

Command	Description
debug list	Filters debugging information on a per-interface or per-access list basis.
debug llc2 dynwind	Displays changes to the dynamic window over Frame Relay.
debug llc2 errors	Displays LLC2 protocol error conditions or unexpected input.
debug llc2 state	Displays state transitions of the LLC2 protocol.

debug llc2 state

To display state transitions of the Logical Link Control, type 2 (LLC2) protocol, use the **debug llc2 state** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug llc2 state

no debug llc2 state

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

Refer to the ISO/IEC standard 8802-2 for definitions and explanations of **debug llc2 state** command output.

Examples

The following is sample output from the **debug llc2 state** command when a router disables and enables an interface:

```
Router# debug llc2 state
LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, NORMAL -> AWAIT (P_TIMER_EXP)
LLC(rs): 0006.f4e1.2a20 0000.303a.90cf 04 04, AWAIT -> D_CONN (P_TIMER_EXP)
LLC: cleanup 0006.f4e1.2a20 0000.303a.90cf 04 04, UNKNOWN (17)
LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, ADM -> SETUP (CONN_REQ)
LLC: normalstate: set_local_busy 0006.f4e1.2a20 0000.303a.90cf 04 04
LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, NORMAL -> BUSY (SET_LOCAL_BUSY)
LLC: Connection established: 0006.f4e1.2a20 0000.303a.90cf 04 04, success
LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, SETUP -> BUSY (SET_LOCAL_BUSY)
LLC: busystate: 0006.f4e1.2a20 0000.303a.90cf 04 04 local busy cleared
LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, BUSY -> NORMAL (CLEAR_LOCAL_BUSY)
```

Related Commands

Command	Description
debug list	Filters debugging information on a per-interface or per-access list basis.
debug llc2 dynwind	Displays changes to the dynamic window over Frame Relay.
debug llc2 errors	Displays LLC2 protocol error conditions or unexpected input.
debug llc2 packet	Displays all input and output from the LLC2 protocol stack.

debug lnm events

To display any unusual events that occur on a Token Ring network, use the **debug lnm events** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug lnm events

no debug lnm events

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

Unusual events include stations reporting errors or error thresholds being exceeded.

Examples

The following is sample output from the **debug lnm events** command:

```
Router# debug lnm events
IBMNM3: Adding 0000.3001.1166 to error list
IBMNM3: Station 0000.3001.1166 going into preweight condition
IBMNM3: Station 0000.3001.1166 going into weight condition
IBMNM3: Removing 0000.3001.1166 from error list
LANMGR0: Beaconing is present on the ring
LANMGR0: Ring is no longer beaconing
IBMNM3: Beaconing, Postmortem Started
IBMNM3: Beaconing, heard from 0000.3000.1234
IBMNM3: Beaconing, Postmortem Next Stage
IBMNM3: Beaconing, Postmortem Finished
```

The following message indicates that station 0000.3001.1166 reported errors and has been added to the list of stations reporting errors. This station is located on Ring 3.

```
IBMNM3: Adding 0000.3001.1166 to error list
```

The following message indicates that station 0000.3001.1166 has passed the “early warning” threshold for error counts:

```
IBMNM3: Station 0000.3001.1166 going into preweight condition
```

The following message indicates that station 0000.3001.1166 is experiencing a severe number of errors:

```
IBMNM3: Station 0000.3001.1166 going into weight condition
```

The following message indicates that the error counts for station 0000.3001.1166 have all decayed to zero, so this station is being removed from the list of stations that have reported errors:

```
IBMNM3: Removing 0000.3001.1166 from error list
```

The following message indicates that Ring 0 has entered failure mode. This ring number is assigned internally.

```
LANMGR0: Beaconing is present on the ring
```

The following message indicates that Ring 0 is no longer in failure mode. This ring number is assigned internally.

```
LANMGR0: Ring is no longer beaconing
```

The following message indicates that the router is beginning its attempt to determine whether any stations left the ring during the automatic recovery process for the last beaconing failure. The router attempts to contact stations that were part of the fault domain to detect whether they are still operating on the ring.

```
IBMNM3: Beaconing, Postmortem Started
```

The following message indicates that the router is attempting to determine whether any stations left the ring during the automatic recovery process for the last beaconing failure. It received a response from station 0000.3000.1234, one of the two stations in the fault domain.

```
IBMNM3: Beaconing, heard from 0000.3000.1234
```

The following message indicates that the router is attempting to determine whether any stations left the ring during the automatic recovery process for the last beaconing failure. It is initiating another attempt to contact the two stations in the fault domain.

```
IBMNM3: Beaconing, Postmortem Next Stage
```

The following message indicates that the router has attempted to determine whether any stations left the ring during the automatic recovery process for the last beaconing failure. It has successfully heard back from both stations that were part of the fault domain.

```
IBMNM3: Beaconing, Postmortem Finished
```

Explanations follow for other messages that the **debug lnm events** command can generate.

The following message indicates that the router is out of memory:

```
LANMGR: memory request failed, find_or_build_station()
```

The following message indicates that Ring 3 is experiencing a large number of errors that cannot be attributed to any individual station:

```
IBMNM3: Non-isolating error threshold exceeded
```

The following message indicates that a station (or stations) on Ring 3 is receiving frames faster than they can be processed:

```
IBMNM3: Adapters experiencing congestion
```

The following message indicates that the beaconing has lasted for over 1 minute and is considered a “permanent” error:

```
IBMNM3: Beaconing, permanent
```

The following message indicates that the beaconing lasted for less than 1 minute. The router is attempting to determine whether either station in the fault domain left the ring.

```
IBMNM: Beaconing, Destination Started
```

In the preceding line of output, the following can replace “Started”: “Next State,” “Finished,” “Timed out,” and “Cannot find station *n*.”

debug lnm llc

To display all comm unication between the router/bridge and the LAN Network Managers (LNMs) that have connections to it, use the **debug lnm llc** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug lnm llc

no debug lnm llc

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

One line is displayed for each message sent or received.

Examples

The following is sample output from the **debug lnm llc** command:

```
Router# debug lnm llc
IBMNM: Received LRM Set Reporting Point frame from 1000.5ade.0d8a.
IBMNM: found bridge: 001-2-00A, addresses: 0000.3040.a630 4000.3040.a630
IBMNM: Opening connection to 1000.5ade.0d8a on TokenRing0
IBMNM: Sending LRM LAN Manager Accepted to 1000.5ade.0d8a on link 0.
IBMNM: sending LRM New Reporting Link Established to 1000.5a79.dbf8 on link 1.
IBMNM: Determining new controlling LNM
IBMNM: Sending Report LAN Manager Control Shift to 1000.5ade.0d8a on link 0.
IBMNM: Sending Report LAN Manager Control Shift to 1000.5a79.dbf8 on link 1.
IBMNM: Bridge 001-2-00A received Request Bridge Status from 1000.5ade.0d8a.
IBMNM: Sending Report Bridge Status to 1000.5ade.0d8a on link 0.
IBMNM: Bridge 001-2-00A received Request REM Status from 1000.5ade.0d8a.
IBMNM: Sending Report REM Status to 1000.5ade.0d8a on link 0.
IBMNM: Bridge 001-2-00A received Set Bridge Parameters from 1000.5ade.0d8a.
IBMNM: Sending Bridge Parameters Set to 1000.5ade.0d8a on link 0.
IBMNM: sending Bridge Params Changed Notification to 1000.5a79.dbf8 on link 1.
IBMNM: Bridge 001-2-00A received Set REM Parameters from 1000.5ade.0d8a.
IBMNM: Sending REM Parameters Set to 1000.5ade.0d8a on link 0.
IBMNM: sending REM Parameters Changed Notification to 1000.5a79.dbf8 on link 1.
IBMNM: Bridge 001-2-00A received Set REM Parameters from 1000.5ade.0d8a.
IBMNM: Sending REM Parameters Set to 1000.5ade.0d8a on link 0.
IBMNM: sending REM Parameters Changed Notification to 1000.5a79.dbf8 on link 1.
IBMNM: Received LRM Set Reporting Point frame from 1000.5ade.0d8a.
IBMNM: found bridge: 001-1-00A, addresses: 0000.3080.2d79 4000.3080.2d7
```

As the output indicates, the **debug lnm llc** command output can vary somewhat in format.

The table below describes the significant fields shown in the display.

Table 22: debug lnm llc Field Descriptions

Field	Description
IBMNM:	Displays LLC-level debugging information.
Received	Router received a frame. The other possible value is Sending, to indicate that the router is sending a frame.

Field	Description
LRM	The function of the LLC-level software that is communicating as follows: • CRS--Configuration Report Server • LBS--LAN Bridge Server • LRM--LAN Reporting Manager • REM--Ring Error Monitor • RPS--Ring Parameter Server • RS--Ring Station
Set Reporting Point	Name of the specific frame that the router sent or received. Possible values include the following: • Bridge Counter Report • Bridge Parameters Changed Notification • Bridge Parameters Set • CRS Remove Ring Station • CRS Report NAUN Change • CRS Report Station Information • CRS Request Station Information • CRS Ring Station Removed • LRM LAN Manager Accepted • LRM Set Reporting Point • New Reporting Link Established • REM Forward MAC Frame • REM Parameters Changed Notification • REM Parameters Set • Report Bridge Status • Report LAN Manager Control Shift • Report REM Status • Request Bridge Status • Request REM Status • Set Bridge Parameters • Set REM Parameters

Field	Description
from 1000.5ade.0d8a	If the router has received the frame, this address is the source address of the frame. If the router is sending the frame, this address is the destination address of the frame.

The following message indicates that the lookup for the bridge with which the LAN Manager was requesting to communicate was successful:

```
IBMNM: found bridge: 001-2-00A, addresses: 0000.3040.a630 4000.3040.a630
```

The following message indicates that the connection is being opened:

```
IBMNM: Opening connection to 1000.5ade.0d8a on TokenRing0
```

The following message indicates that a LAN Manager has connected or disconnected from an internal bridge and that the router computes which LAN Manager is allowed to change parameters:

```
IBMNM: Determining new controlling LNM
```

The following line of output indicates which bridge in the router is the destination for the frame:

```
IBMNM: Bridge 001-2-00A received Request Bridge Status from 1000.5ade.0d8a.
```

debug lnm mac

To display all management communication between the router/bridge and all stations on the local Token Rings, use the **debug lnm mac** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug lnm mac

no debug lnm mac

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

One line is displayed for each message sent or received.

Examples

The following is sample output from the **debug lnm mac** command:

```
Router# debug lnm mac
LANMGR0: RS received request address from 4000.3040.a670.
LANMGR0: RS sending report address to 4000.3040.a670.
LANMGR0: RS received request state from 4000.3040.a670.
LANMGR0: RS sending report state to 4000.3040.a670.
LANMGR0: RS received request attachments from 4000.3040.a670.
LANMGR0: RS sending report attachments to 4000.3040.a670.
LANMGR2: RS received ring purge from 0000.3040.a630.
LANMGR2: CRS received report NAUN change from 0000.3040.a630.
LANMGR2: RS start watching ring poll.
LANMGR0: CRS received report NAUN change from 0000.3040.a630.
LANMGR0: RS start watching ring poll.
LANMGR2: REM received report soft error from 0000.3040.a630.
LANMGR0: REM received report soft error from 0000.3040.a630.
LANMGR2: RS received ring purge from 0000.3040.a630.
LANMGR2: RS received AMP from 0000.3040.a630.
LANMGR2: RS received SMP from 0000.3080.2d79.
LANMGR2: CRS received report NAUN change from 1000.5ade.0d8a.
LANMGR2: RS start watching ring poll.
LANMGR0: RS received ring purge from 0000.3040.a630.
LANMGR0: RS received AMP from 0000.3040.a630.
LANMGR0: RS received SMP from 0000.3080.2d79.
LANMGR0: CRS received report NAUN change from 1000.5ade.0d8a.
LANMGR0: RS start watching ring poll.
LANMGR2: RS received SMP from 1000.5ade.0d8a.
LANMGR2: RPS received request initialization from 1000.5ade.0d8a.
LANMGR2: RPS sending initialize station to 1000.5ade.0d8a.
```

The table below describes the significant fields shown in the display.

Table 23: debug lnm mac Field Descriptions

Field	Description
LANMGR0:	Indicates that this line of output displays MAC-level debugging information. 0 indicates the number of the Token Ring interface associated with this line of debugging output.
RS	Indicates which function of the MAC-level software is communicating as follows: • CRS--Configuration Report Server • REM--Ring Error Monitor • RPS--Ring Parameter Server • RS--Ring Station
received	Indicates that the router received a frame. The other possible value is sending, to indicate that the router is sending a frame.
request address	Indicates the name of the specific frame that the router sent or received. Possible values include the following: • AMP • initialize station • report address • report attachments • report nearest active upstream neighbor (NAUN) change • report soft error • report state • request address • request attachments • request initialization • request state • ring purge • SMP

Field	Description
from 4000.3040.a670	Indicates the source address of the frame, if the router has received the frame. If the router is sending the frame, this address is the destination address of the frame.

As the output indicates, all **debug lnm mac** command messages follow the format described in the table above except the following:

```
LANMGR2: RS start watching ring poll
LANMGR2: RS stop watching ring poll
```

These messages indicate that the router starts and stops receiving AMP and SMP frames. These frames are used to build a current picture of which stations are on the ring.

debug local-ack state

To display the new and the old state conditions whenever there is a state change in the local acknowledgment state machine, use the **debug loc al-ack state** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug local-ack state

no debug local-ack state

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Examples

The following is sample output from the **debug local-ack state** command:

```
Router# debug local-ack state
LACK_STATE: 2370300, hashp 2AE628, old state = disconn, new state = awaiting
LLC2 open to finish
LACK_STATE: 2370304, hashp 2AE628, old state = awaiting LLC2 open to finish,
new state = connected
LACK_STATE: 2373816, hashp 2AE628, old state = connected, new state = disconnected
LACK_STATE: 2489548, hashp 2AE628, old state = disconn, new state = awaiting
LLC2 open to finish
LACK_STATE: 2489548, hashp 2AE628, old state = awaiting LLC2 open to finish,
new state = connected
LACK_STATE: 2490132, hashp 2AE628, old state = connected, new state = awaiting
linkdown response
LACK_STATE: 2490140, hashp 2AE628, old state = awaiting linkdown response,
new state = disconnected
LACK_STATE: 2497640, hashp 2AE628, old state = disconn, new state = awaiting
LLC2 open to finish
LACK_STATE: 2497644, hashp 2AE628, old state = awaiting LLC2 open to finish,
new state = connected
```

The table below describes the significant fields shown in the display.

Table 24: debug local-ack state Field Descriptions

Field	Description
LACK_STATE:	Indicates that this packet describes a state change in the local acknowledgment state machine.
2370300	System clock.
hashp 2AE628	Internal control block pointer used by technical support staff for debugging purposes.

Field	Description
old state = disconn	Old state condition in the local acknowledgment state machine. Possible values include the following: • Disconn (disconnected) • awaiting LLC2 open to finish • connected • awaiting linkdown response
new state = awaiting LLC2 open to finish	New state condition in the local acknowledgment state machine. Possible values include the following: • Disconn (disconnected) • awaiting LLC2 open to finish • connected • awaiting linkdown response